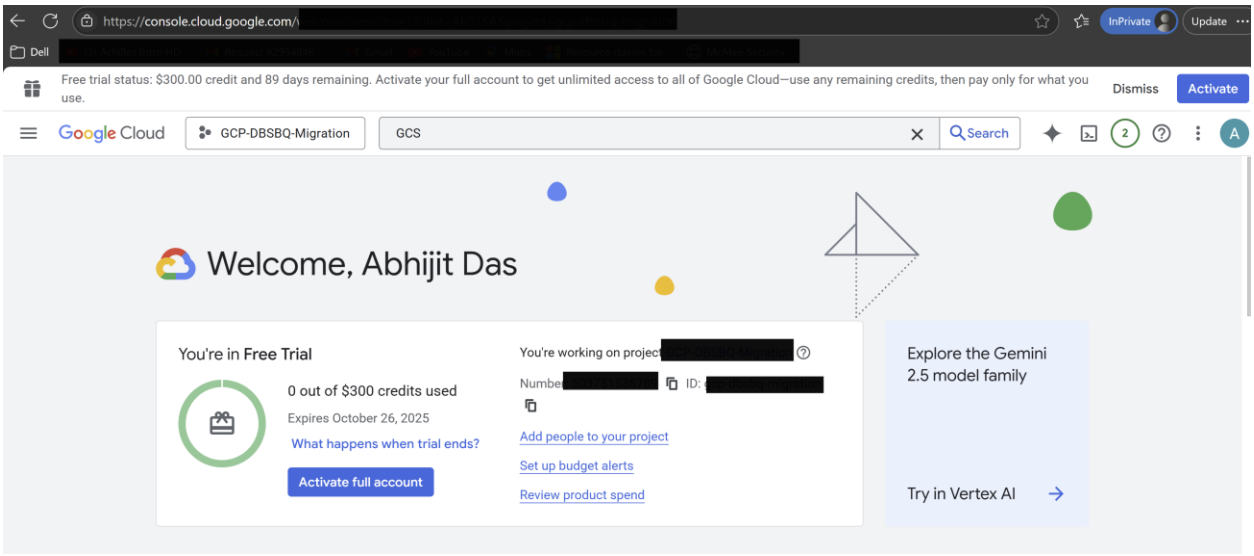
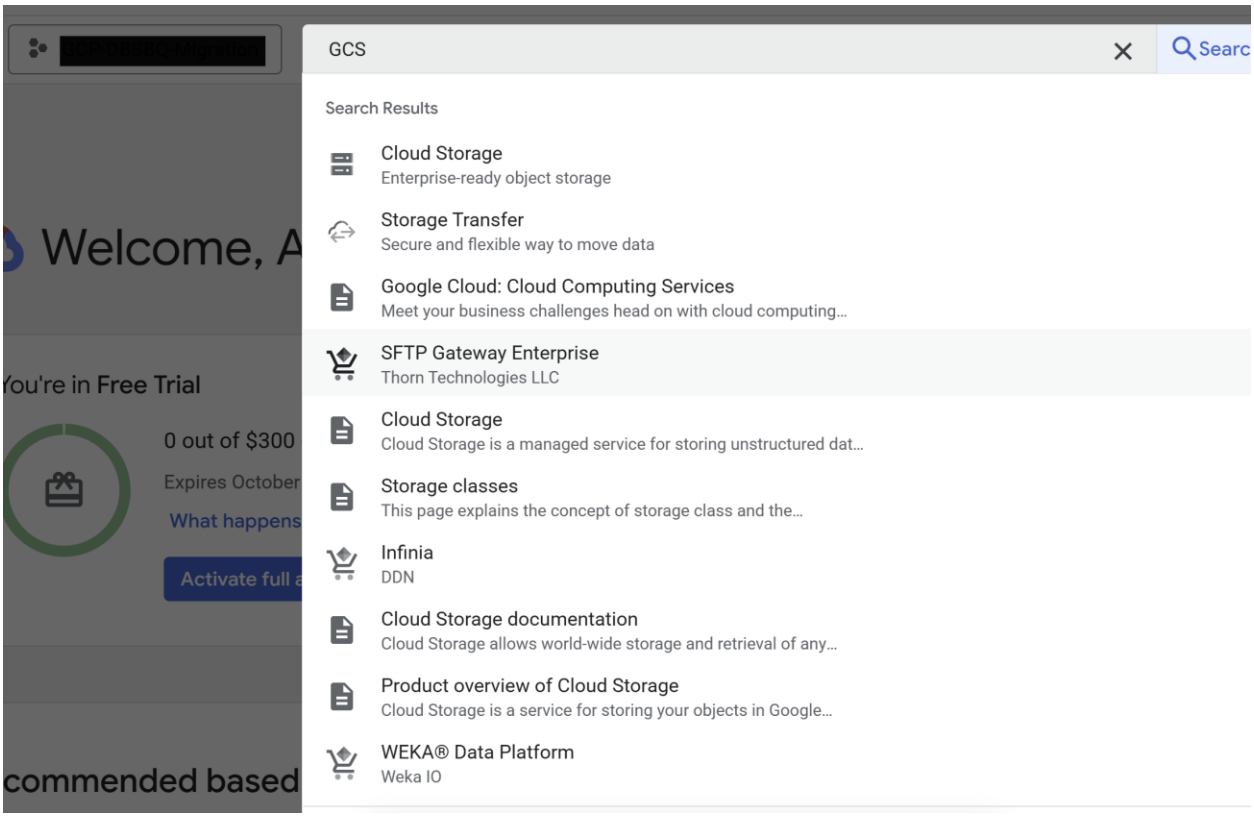
# Pre-requisites

1. Your data is already in GCS (e.g., CSV, JSON, or Parquet format).
2. You have a **BigQuery dataset** created (e.g., `my_dataset`).
3. The BigQuery service account has **read access to the GCS bucket**.

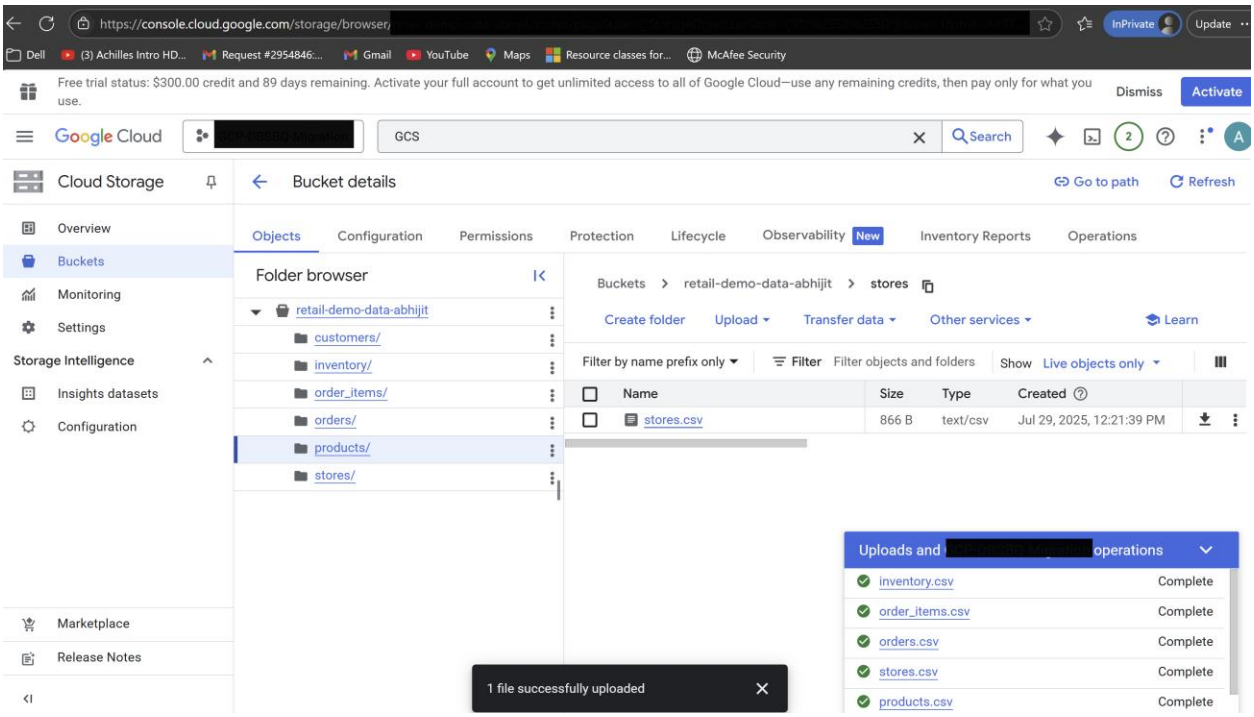
Create Google Cloud Account:



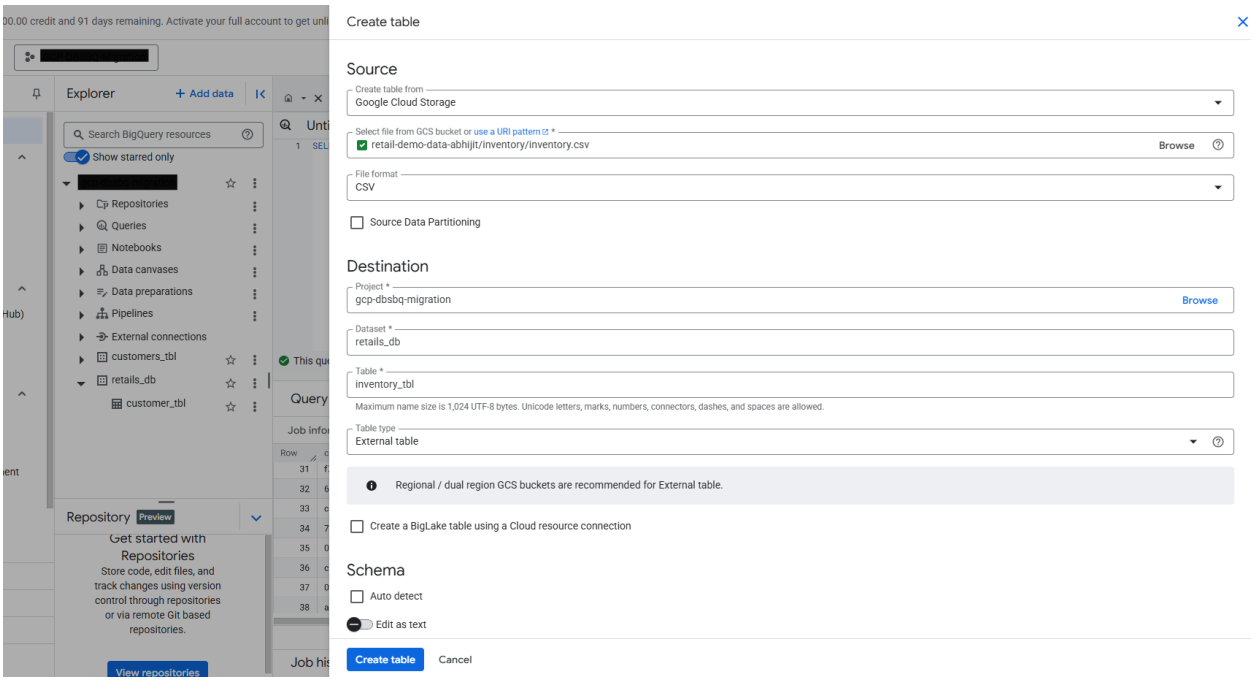

Create buckets

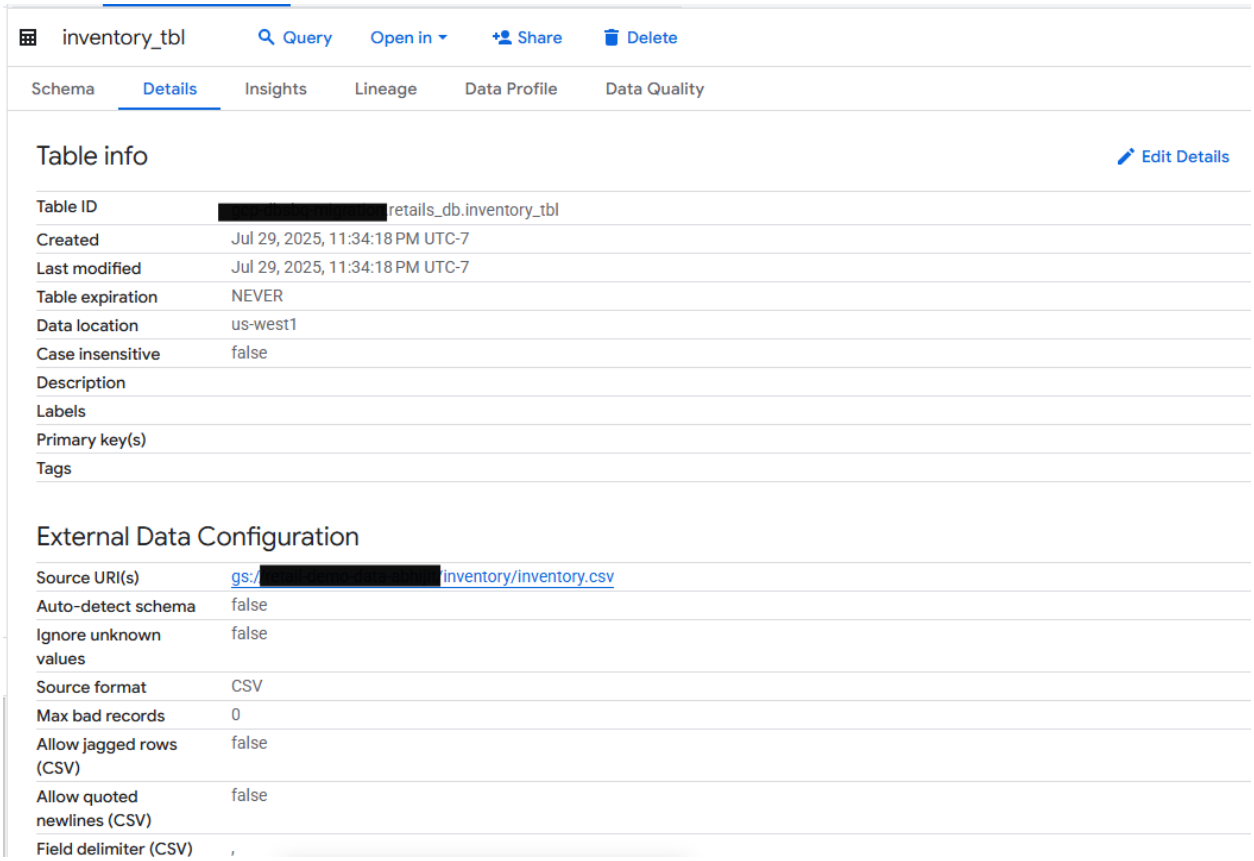



Create GCS bucket and upload files to folders in buckets

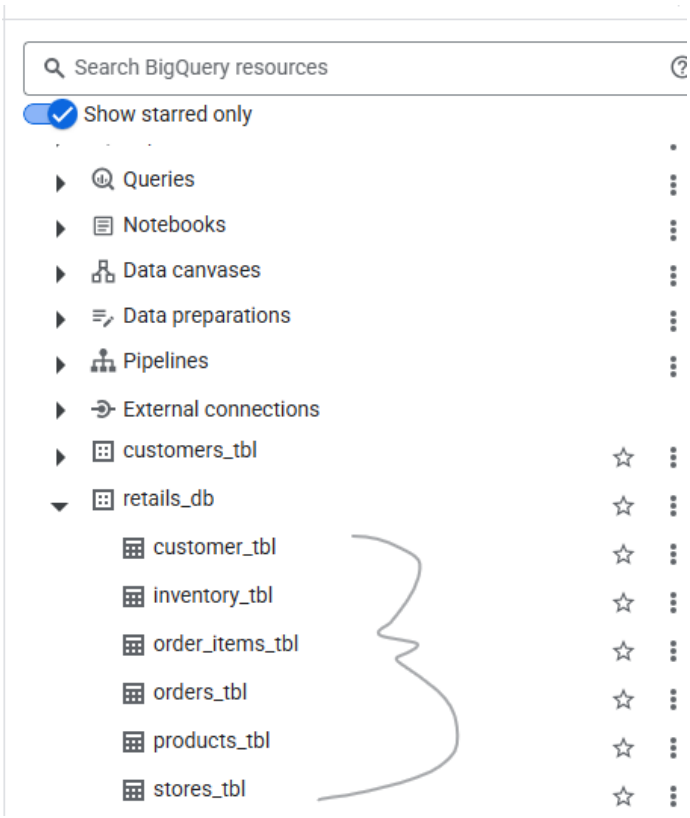



Now we need to update GCS bucket → Permissions to access BigQuery Data - BigQuery Data Editor / BigQuery Job User



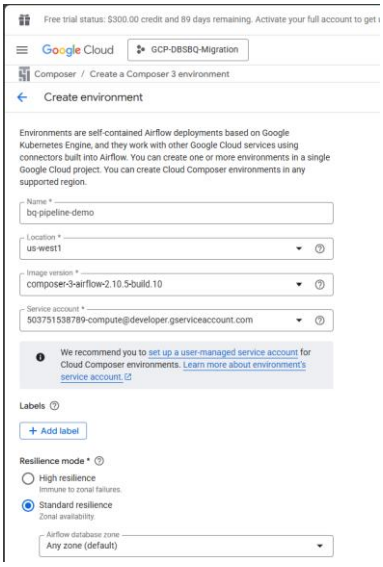



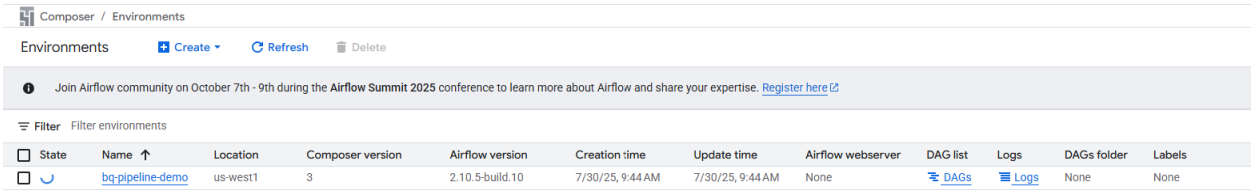

Load remaining files to BigQuery tables.



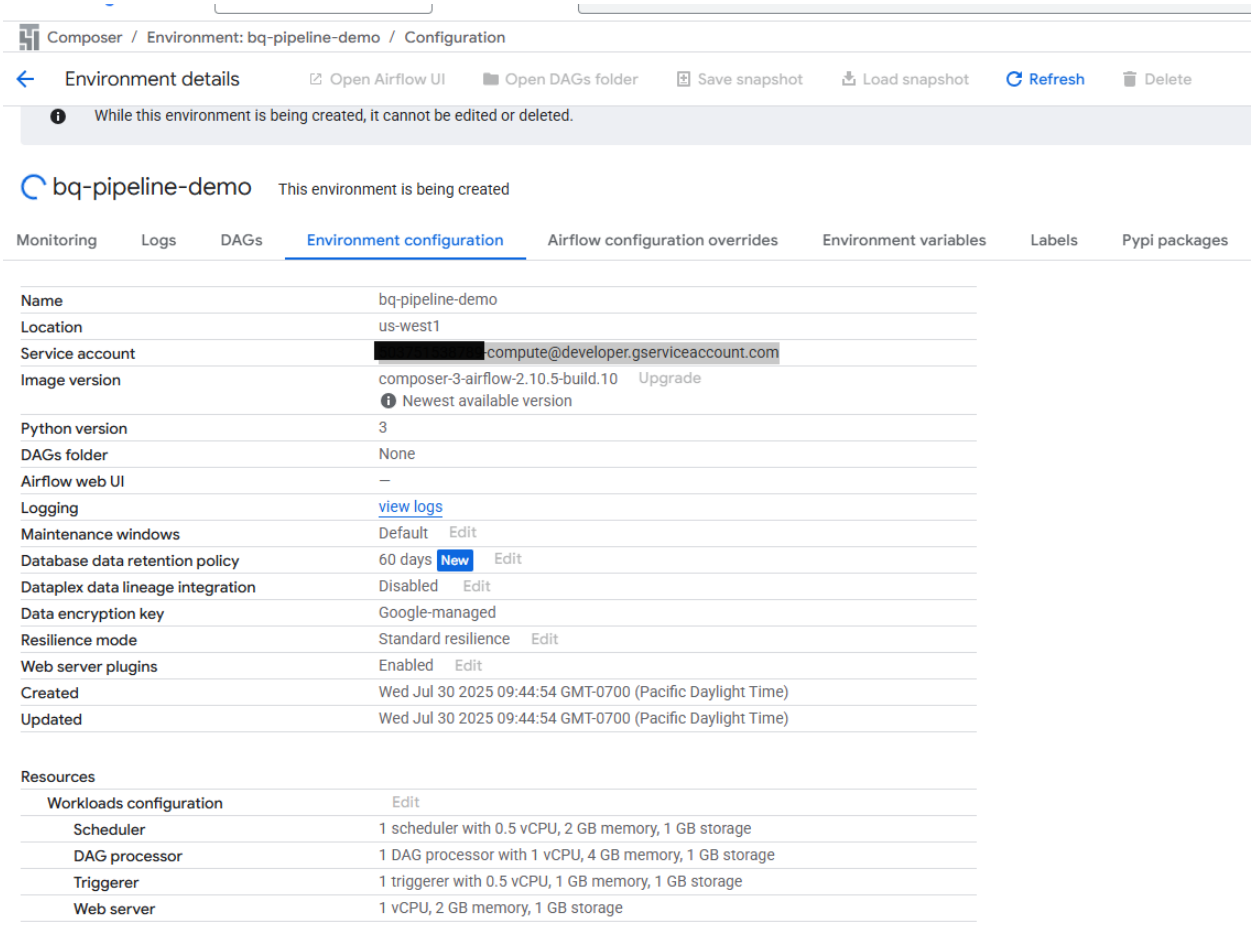

Integration with Composer (GCP) and BigQuery

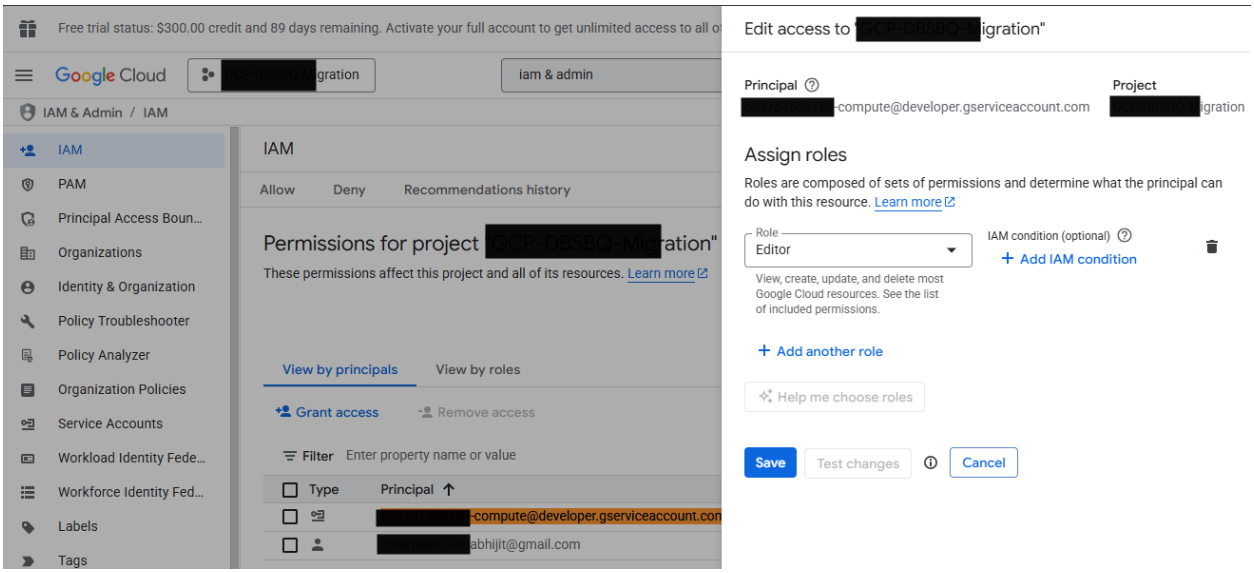
Create Composer environment






## Grant IAM Permissions to Composer's Service Account




Free trial status: $300.00 credit and 89 days remaining. Activate your full account to get unlimited access to all o
Google Cloud
[redacted]gration
iam & admin
IAM & Admin / IAM
IAM
PAM
Principal Access Boun...
Organizations
Identity & Organization
Policy Troubleshooter
Policy Analyzer
Organization Policies
Service Accounts
Workload Identity Fede...
Workforce Identity Fed...
Labels
Tags
IAM
Allow
Deny
Recommendations history
Permissions for project [redacted]ration"
These permissions affect this project and all of its resources. Learn more
View by principals
View by roles
Grant access
Remove access
Filter
Enter property name or value
Type
Principal
[redacted]-compute@developer.gserviceaccount.com
[redacted]abhijit@gmail.com
Edit access to "[redacted]igration"
Principal
[redacted]-compute@developer.gserviceaccount.com
Project
[redacted]igration
Assign roles
Roles are composed of sets of permissions and determine what the principal can do with this resource. Learn more
Role
Editor
View, create, update, and delete most Google Cloud resources. See the list of included permissions.
IAM condition (optional)
Add IAM condition
Add another role
Help me choose roles
Save
Test changes
Cancel

Edit access to "GCP-DBSBQ-Migration"

Principal [redacted]mpute@developer.gserviceaccount.com

Project [redacted]igration

## Assign roles

Roles are composed of sets of permissions and determine what the principal can do with this resource. Learn more

Role: Editor
View, create, update, and delete most Google Cloud resources. See the list of included permissions.
IAM condition (optional) + Add IAM condition

Role: BigQuery Job User
Access to run jobs
IAM condition (optional) + Add IAM condition

Role: BigQuery Admin
Administer all BigQuery resources and data
IAM condition (optional) + Add IAM condition

Role: Storage Object Viewer
Grants access to view objects and their metadata, excluding ACLs. Can also list the objects in a bucket.
IAM condition (optional) + Add IAM condition

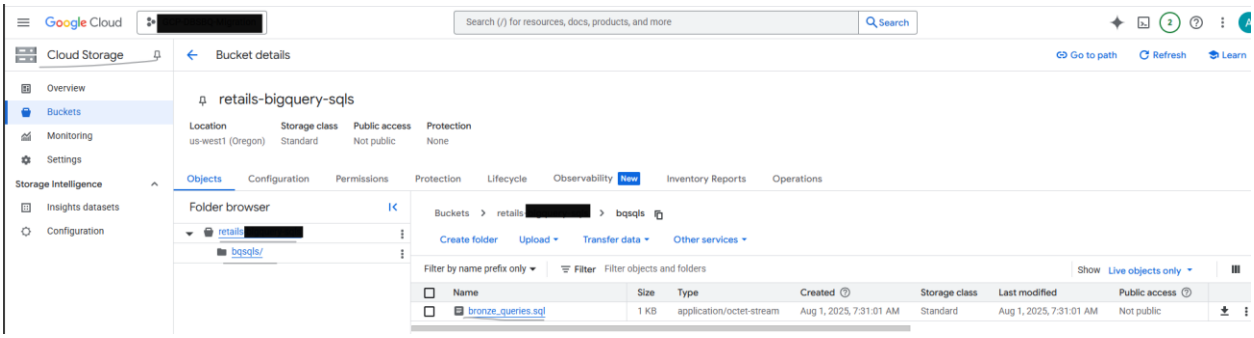
Role: BigQuery Data Editor
Access to edit all the contents of datasets
IAM condition (optional) + Add IAM condition

+ Add another role

Help me choose roles

Save | Test changes | Cancel

Create bucket and upload sql files which we need to run by using Airflow DAGs




Script To run (DAG)

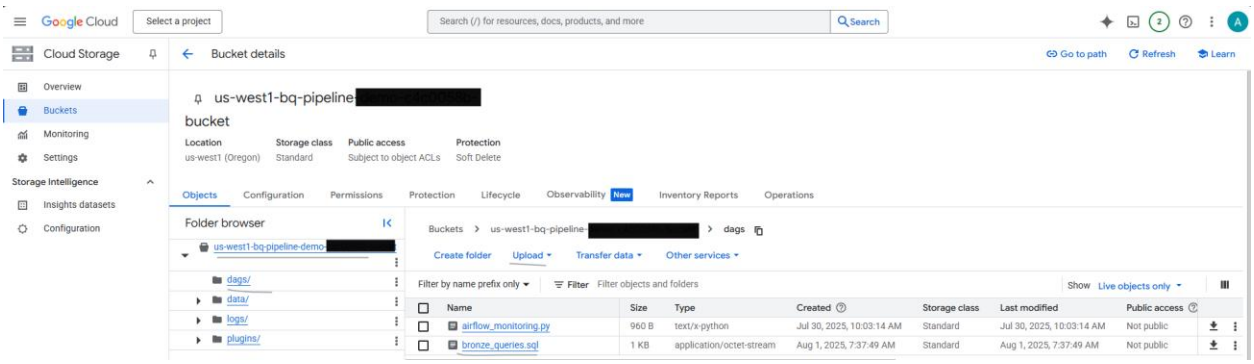
```
from airflow import models
from airflow.providers.google.cloud.operators.bigquery import BigQueryInsertJobOperator
from airflow.utils.dates import days_ago

PROJECT_ID = "your-project-id"
BUCKET_NAME = "retails-[redacted]"
SQL_FILE = "[redacted]bronze_queries.sql"
DATASET = "retail_bronze"
LOCATION = "US"  # or EU, asia-east1, etc.

with models.DAG(
    dag_id="trigger_bq_sql_from_gcs",
    schedule_interval=None,  # Trigger manually or via another DAG
    start_date=days_ago(1),
    catchup=False,
    tags=["bigquery", "gcs"],
) as dag:

    bq_query_from_gcs = BigQueryInsertJobOperator(
        task_id="run_bq_sql_from_gcs",
        configuration={
            "query": {
                "query": f"gs://{BUCKET_NAME}/{SQL_FILE}",
                "useLegacySql": False,
            }
        },
        location=LOCATION,
        project_id=PROJECT_ID,
    )
```

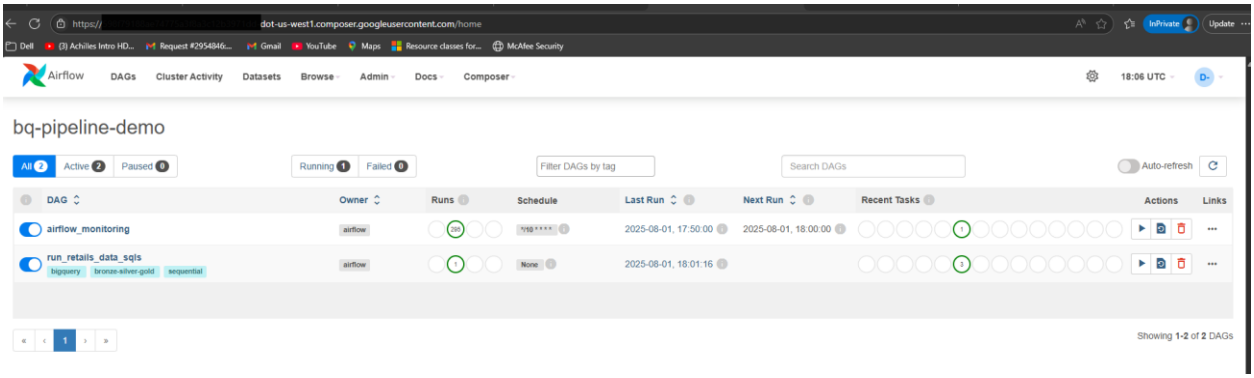
Or we can hard-code sql queries in DAG python script.

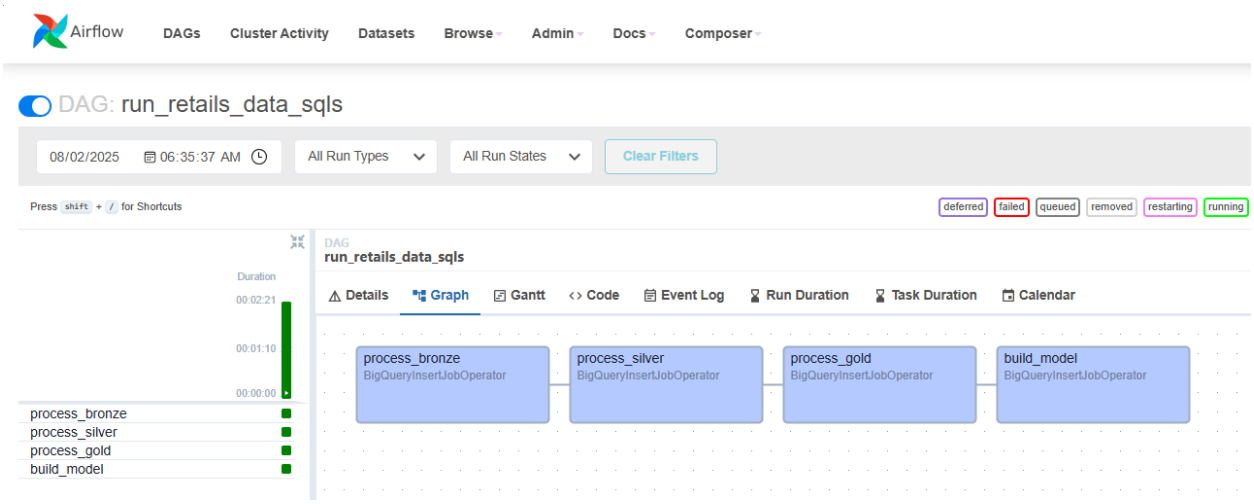
```
from airflow import models
from airflow.providers.google.cloud.operators.bigquery import BigQueryInsertJobOperator
from airflow.utils.dates import days_ago

PROJECT_ID = "gcp-dbsbq-migration"
LOCATION = "us-west1"

# SQL for Bronze
SQL_QUERY_Bronze = """
--Create bronze tables with schema
CREATE SCHEMA IF NOT EXISTS `gcp-dbsbq-migration.retail_bronze`;


-- customers
CREATE OR REPLACE TABLE `gcp-dbsbq-migration.retail_bronze.customers_tbl`
AS SELECT * FROM `gcp-dbsbq-migration.retails_db.customer_tbl`;

--inventory
CREATE OR REPLACE TABLE `gcp-dbsbq-migration.retail_bronze.inventory_tbl`
AS SELECT * FROM `gcp-dbsbq-migration.retails_db.inventory_tbl`;

-- order_items
CREATE OR REPLACE TABLE `gcp-dbsbq-migration.retail_bronze.order_items_tbl`
AS SELECT * FROM `gcp-dbsbq-migration.retails_db.order_items_tbl`;

--orders
CREATE OR REPLACE TABLE `gcp-dbsbq-migration.retail_bronze.orders_tbl`
AS SELECT * FROM `gcp-dbsbq-migration.retails_db.orders_tbl`;

```

```
) as dag:
    silver_task = BigQueryInsertJobOperator(
        configuration={
        },
        location=LOCATION,
        project_id=PROJECT_ID,
    )

    gold_task = BigQueryInsertJobOperator(
        task_id="process_gold",
        configuration={
            "query": {
                "query": SQL_QUERY_Gold,
                "useLegacySql": False
            }
        },
        location=LOCATION,
        project_id=PROJECT_ID,
    )

    ml_model_task = BigQueryInsertJobOperator(
        task_id="build_model",
        configuration={
            "query": {
                "query": SQL_QUERY_MLModel,
                "useLegacySql": False
            }
        },
        location=LOCATION,
        project_id=PROJECT_ID,
    )

    # Set execution order: Bronze -> Silver -> Gold
    bronze_task >> silver_task >> gold_task >> ml_model_task
```

Upload python scripts (DAGs) in Airflow - > Composer - > DAG Folder






Build and run ML model on BigQuery

```sql
--Create ML Models

CREATE OR REPLACE MODEL `[redacted]migration.retail_gold.ml_churn_model`
OPTIONS(model_type='logistic_reg') AS
SELECT
  *,
  CASE WHEN days_since_last_order > 60 THEN 1 ELSE 0 END AS label
FROM `[redacted]migration.retail_gold.gold_customer_features`;


SELECT
  customer_id,
  predicted_label,
  predicted_label_probs
FROM
  ML.PREDICT(MODEL `[redacted]migration.retail_gold.ml_churn_model`,
             (SELECT * FROM `[redacted]migration.retail_gold.gold_customer_features`));
```

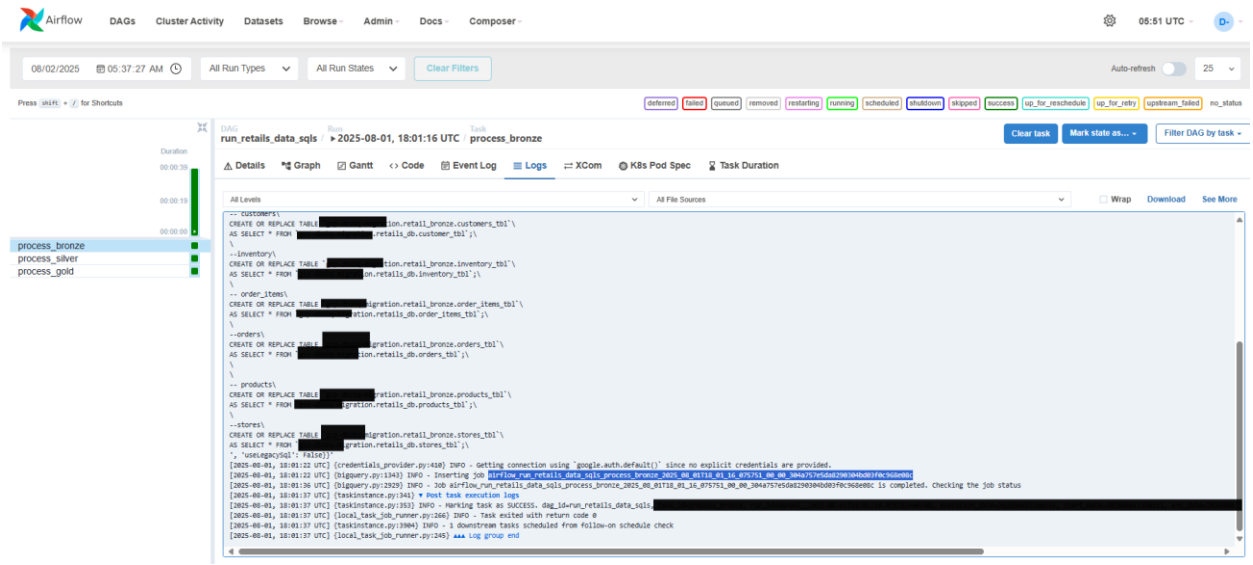
Query completed

Query results

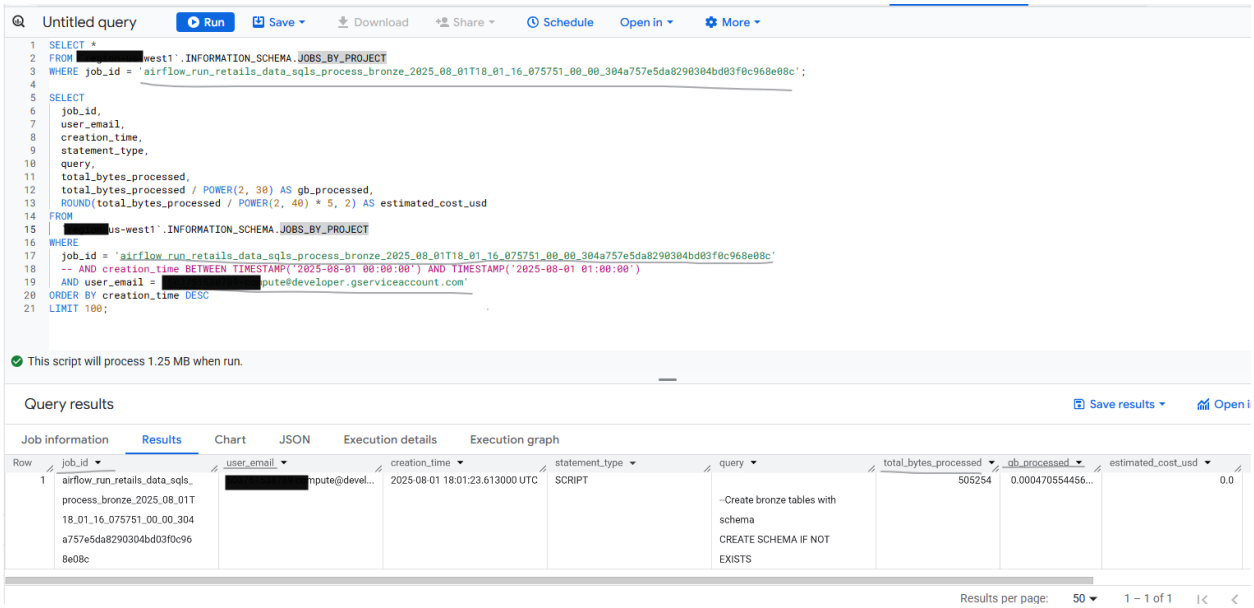
Job information | Results | Chart | JSON | Execution details | Execution graph

| ow | customer_id | predicted_label | predicted_label_probs. label | predicted_label_p... |
|---|---|---|---|---|
| 1 | d3dec6a7-5d04-4b04-a732-3fc3cd961148 | 1 | 1 | 0.999999872247... |
| | | | 0 | 1.277521007869... |
| 2 | f76ba400-93fa-46a6-b91a-927cf0e05ef8 | 1 | 1 | 0.999999889427... |
| | | | 0 | 1.105724314331... |
| 3 | fab12427-b911-46f5-b230-d659a54c464e | 1 | 1 | 0.999999727783... |
| | | | 0 | 2.722164066382... |
| 4 | b3209e23-fcdc-4af0-9a12-ebfe383b5dbe | 0 | 1 | 6.857382860765... |

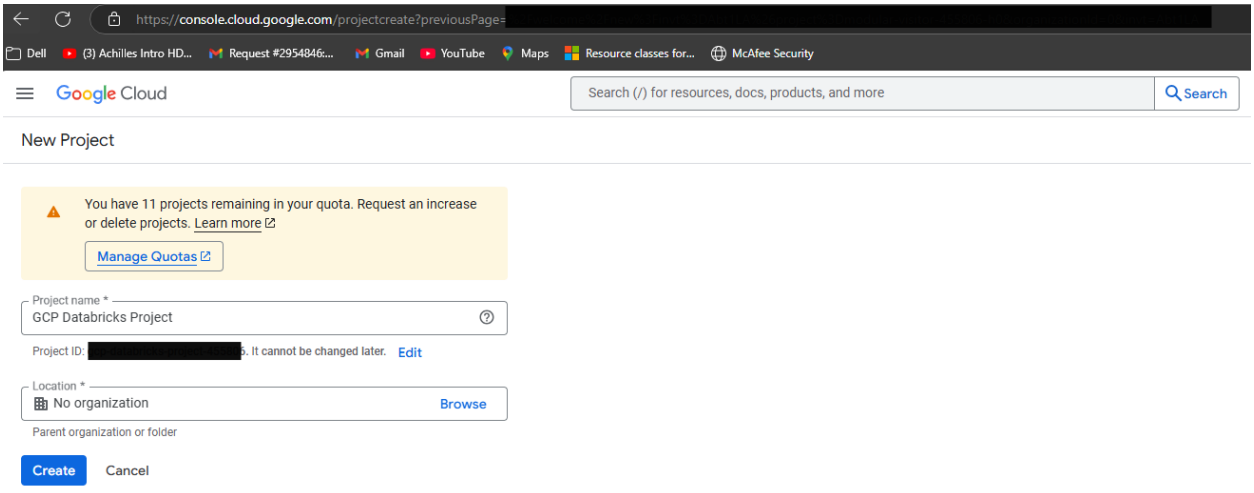
To capture the cost of BigQuery sqls which are triggered by Airflow, we need to find the job_id from the Airflow logs.

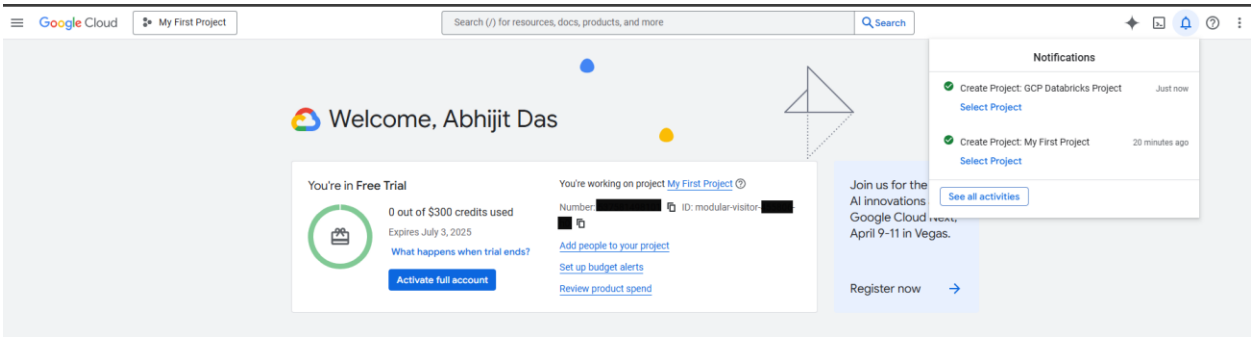


Then we can run following queries in BigQuery to estimate cost. Below screenshot shows only one job_id cost.



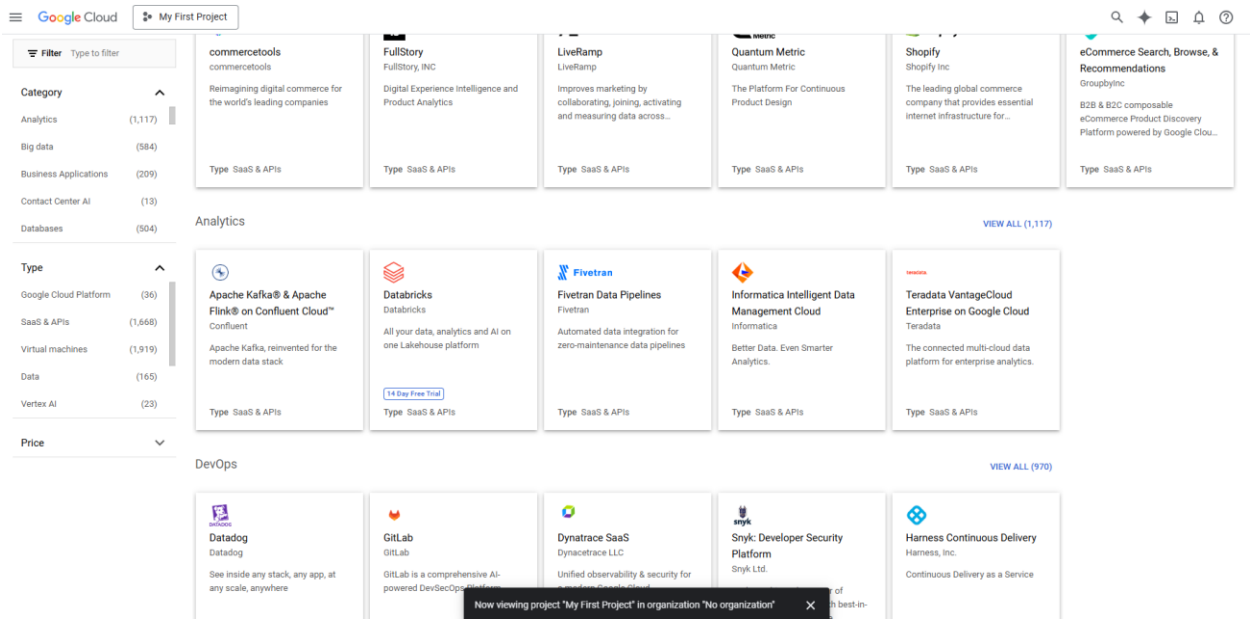

Create a separate project for implementing Databricks Project.



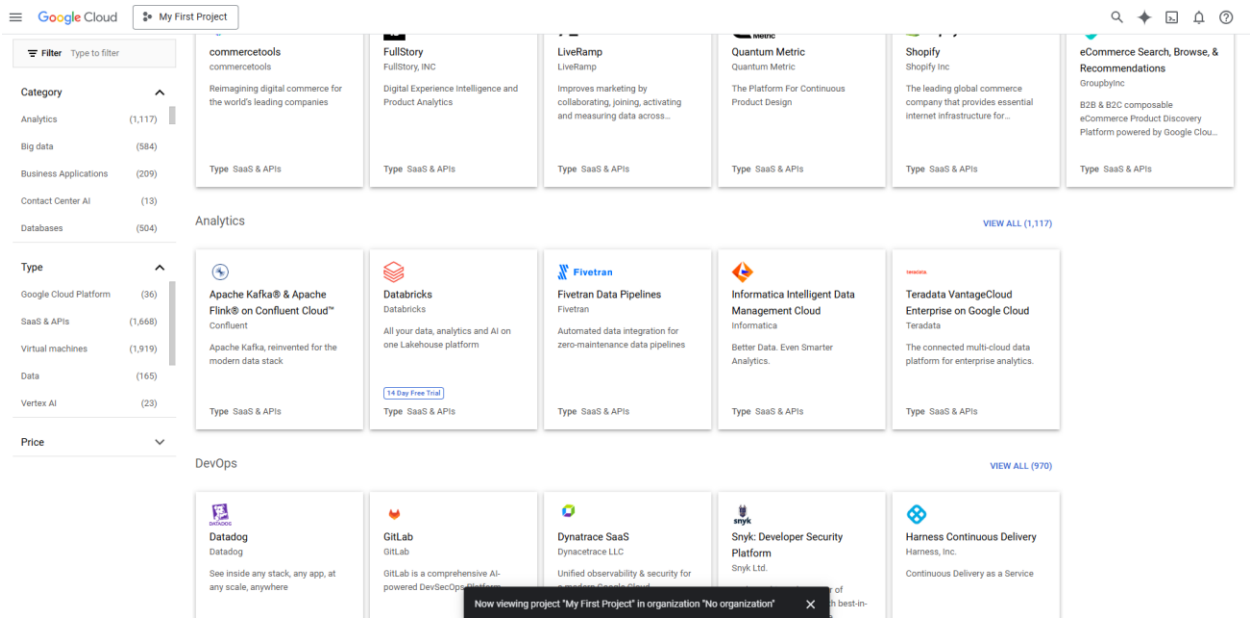



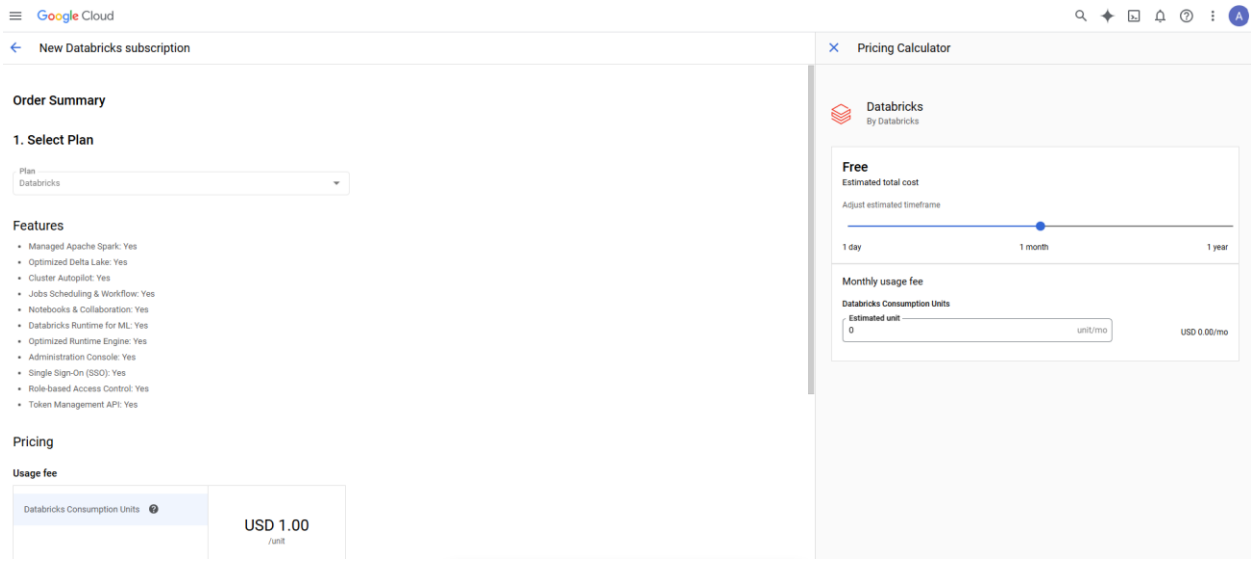
To create GCP Databricks account, we need to subscribe to Databricks.

Find Databricks under Marketplace (Partner Solutions)




Subscribe






## 2. Purchase details

Select a billing account *
My Billing Account

## 3. Terms

**Cancellation and change policy**

- Your subscription fee is billed every month.

**Additional terms**

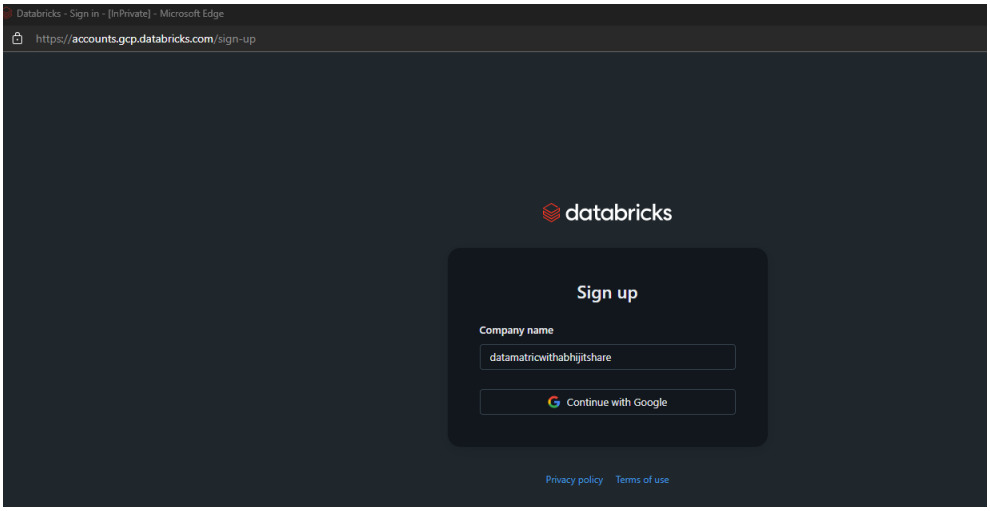
By purchasing, deploying, accessing, or using this product, you acknowledge that Google or an affiliate is the Vendor's agent with respect to this transaction and you agree to comply with the Google Cloud Marketplace Terms of Service (including the GPC terms set forth in Appendix A of the Marketplace ToS), Databricks Terms of Service and the terms of applicable open source software licenses bundled with the product.

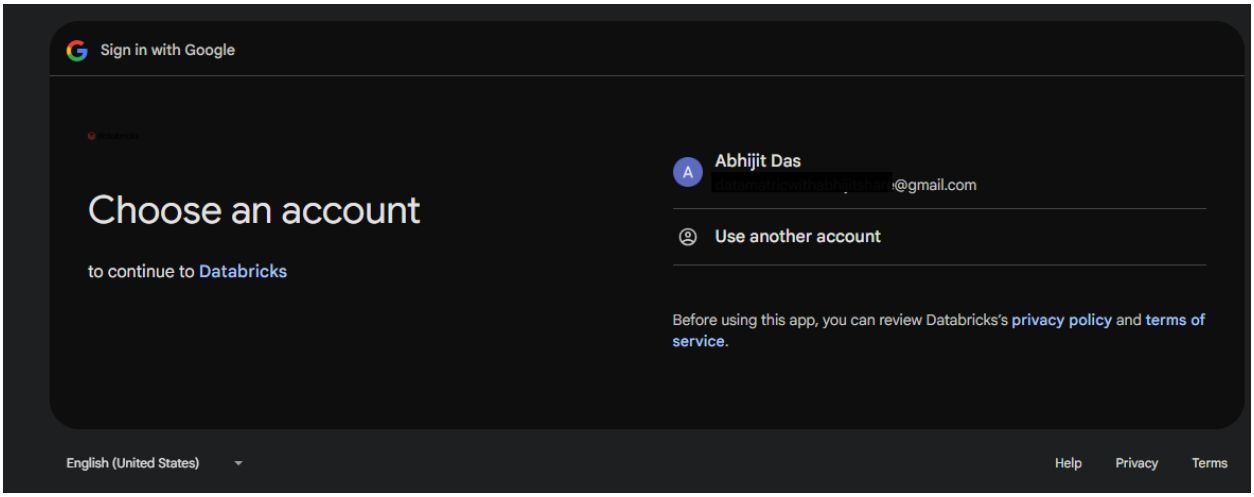
SUBSCRIBE

## Your order request has been sent to Databricks

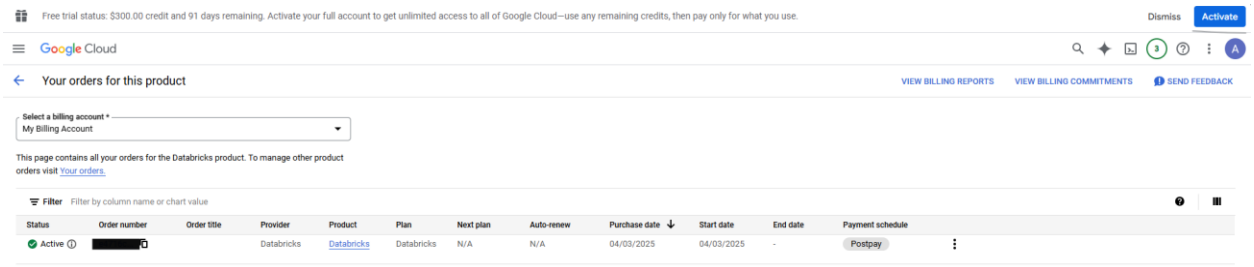
Your new subscription to the Databricks plan for Databricks requires your registration with Databricks. After Databricks approves your request, your subscription will be active and you will begin getting charged. This processing time will depend on the vendor, and you should reach out to Databricks directly with any questions related to signup.

MANAGE ORDERS SIGN UP WITH DATABRICKS

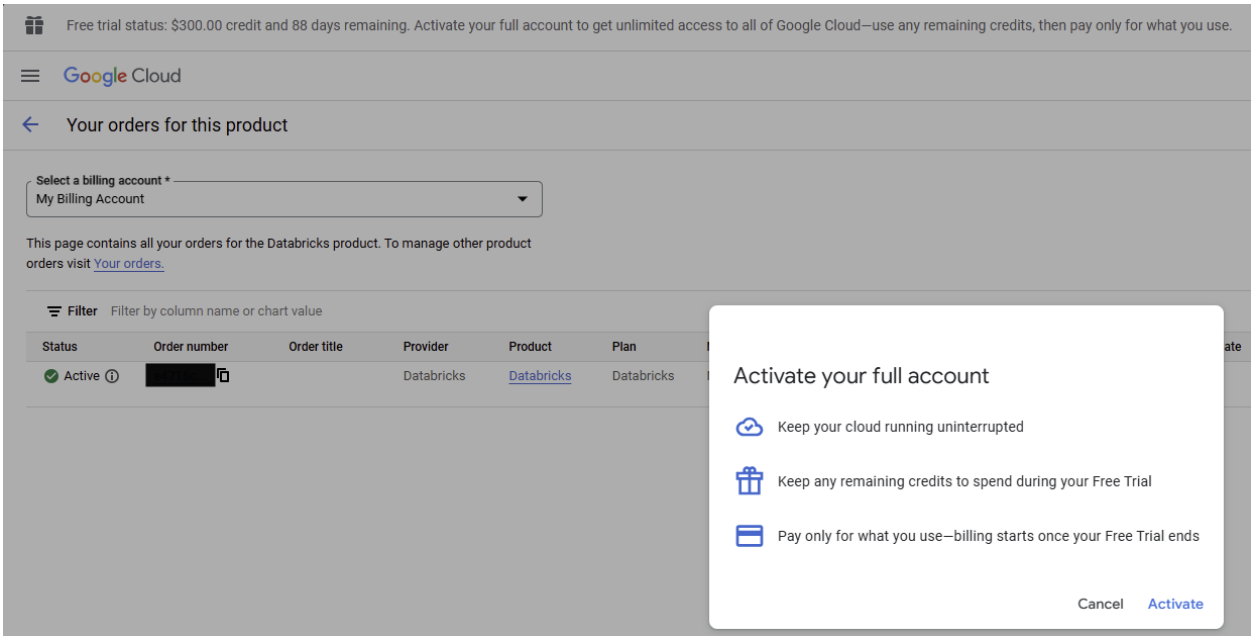


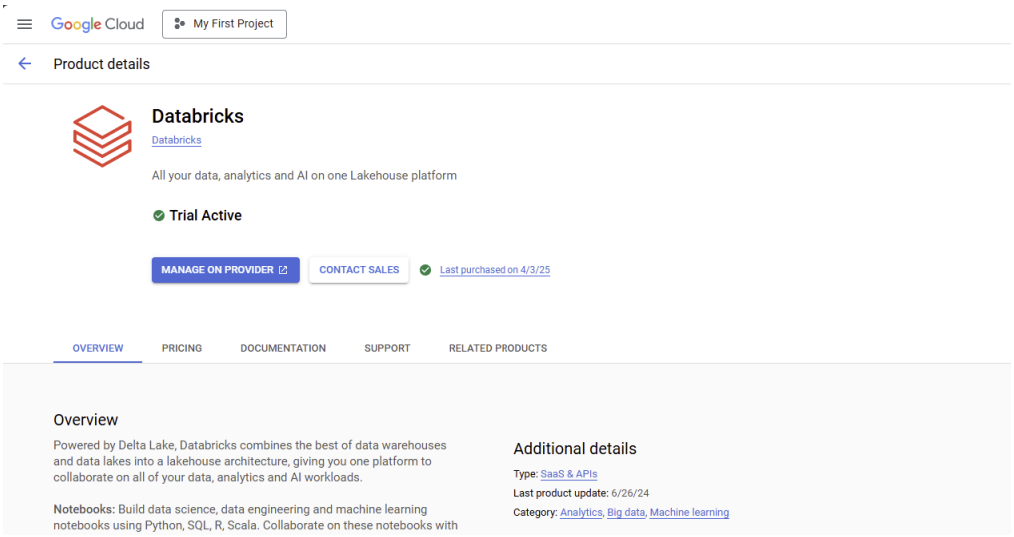



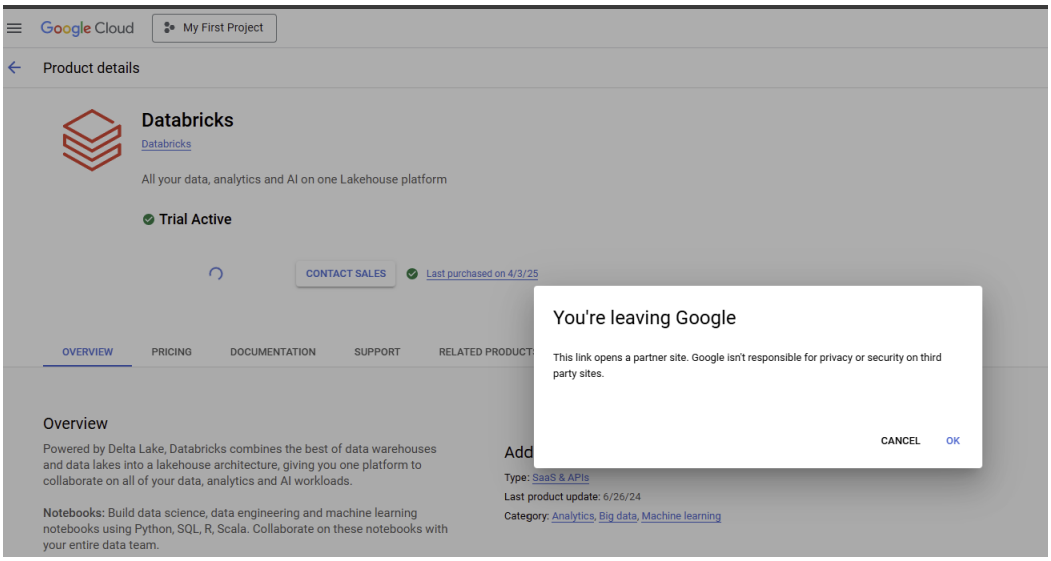

Click on Activate



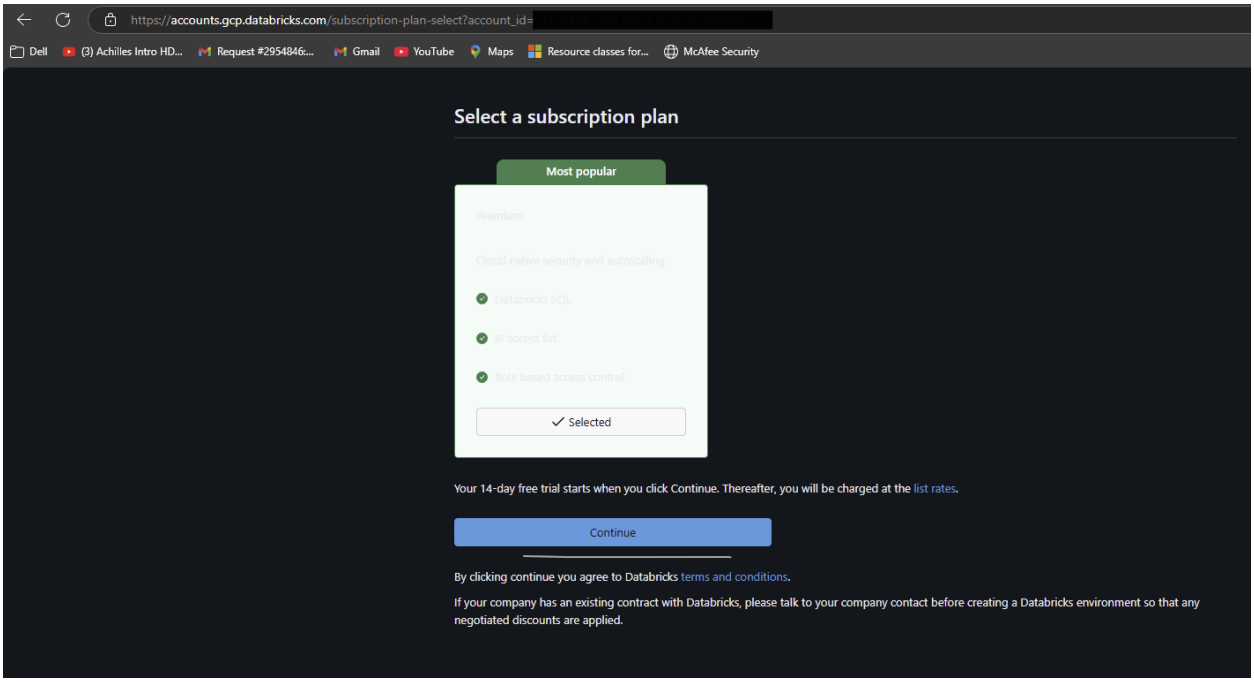

No need to activate the full account, unless you want to use any specific services.

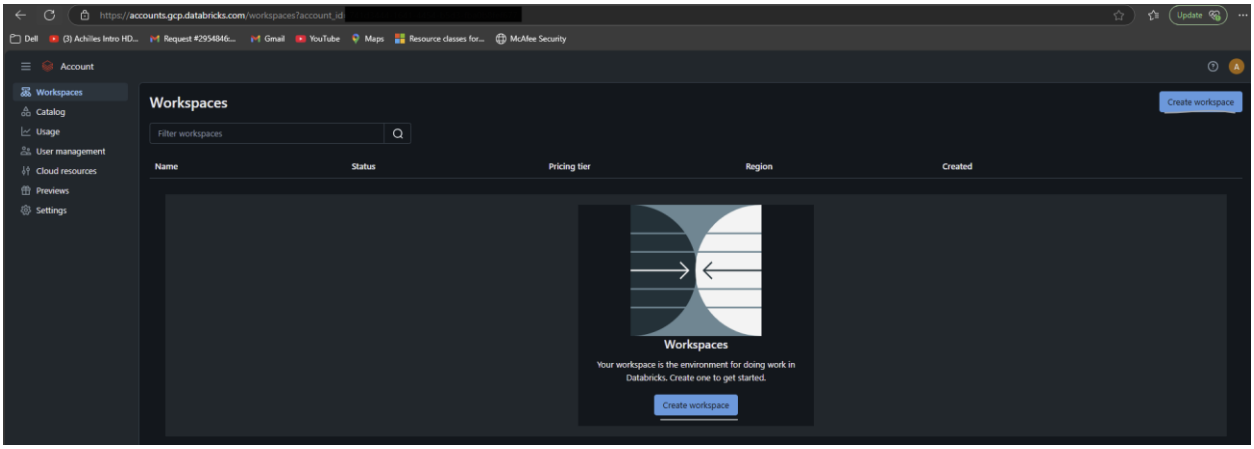

Free trial status: $300.00 credit and 88 days remaining. Activate your full account to get unlimited access to all of Google Cloud—use any remaining credits, then pay only for what you use.
Google Cloud
Your orders for this product
Select a billing account *
My Billing Account
This page contains all your orders for the Databricks product. To manage other product orders visit Your orders.
Filter
Filter by column name or chart value
Status
Order number
Order title
Provider
Product
Plan
Active
[redacted]
Databricks
Databricks
Databricks
Activate your full account
Keep your cloud running uninterrupted
Keep any remaining credits to spend during your Free Trial
Pay only for what you use—billing starts once your Free Trial ends
Cancel
Activate


Google Cloud
My First Project
Product details
Databricks
Databricks
All your data, analytics and AI on one Lakehouse platform
Trial Active
MANAGE ON PROVIDER
CONTACT SALES
Last purchased on 4/3/25
OVERVIEW
PRICING
DOCUMENTATION
SUPPORT
RELATED PRODUCTS
Overview
Powered by Delta Lake, Databricks combines the best of data warehouses and data lakes into a lakehouse architecture, giving you one platform to collaborate on all of your data, analytics and AI workloads.
Notebooks: Build data science, data engineering and machine learning notebooks using Python, SQL, R, Scala. Collaborate on these notebooks with
Additional details
Type: SaaS & APIs
Last product update: 6/26/24
Category: Analytics, Big data, Machine learning

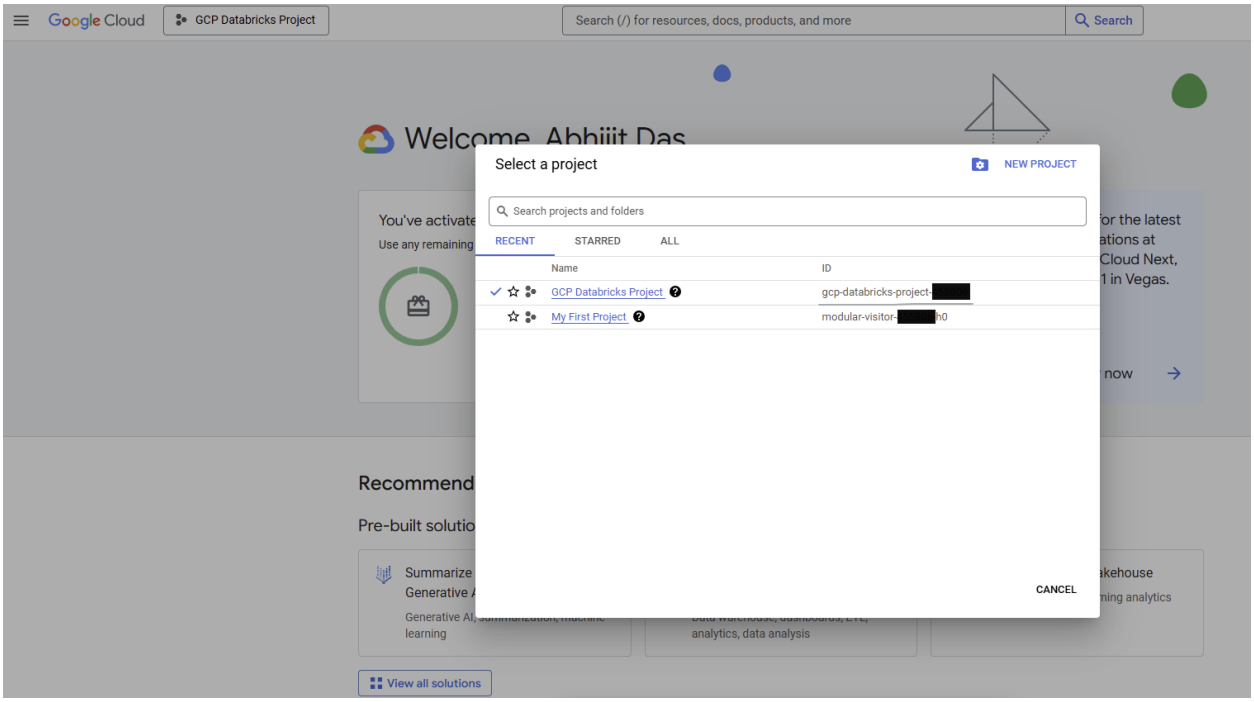


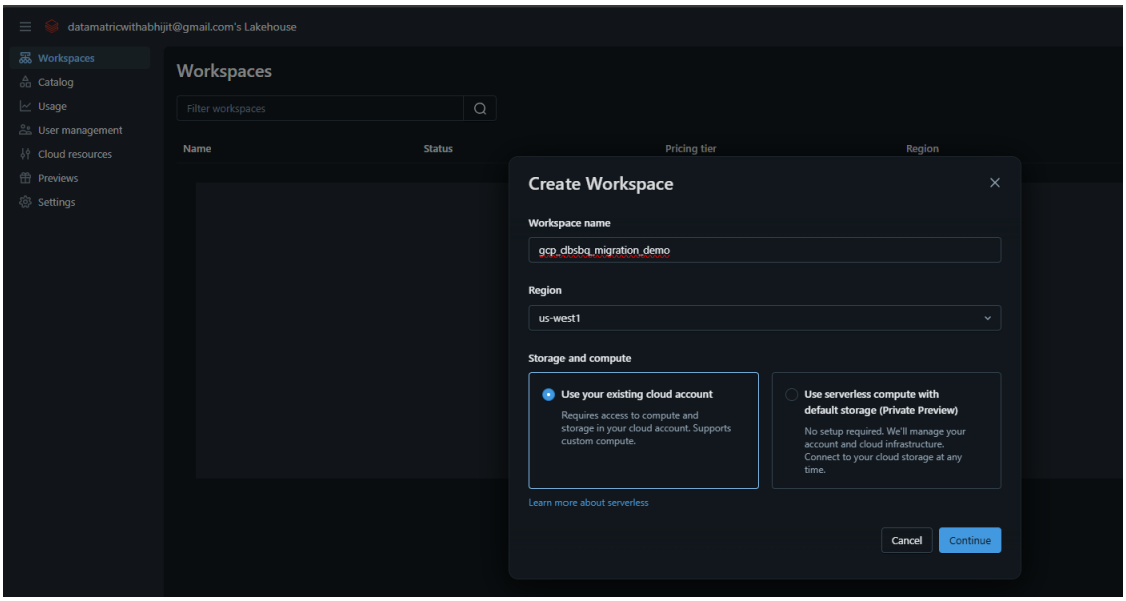

Click on “Continue With Google” account that we used for Databricks account registration and select the plan

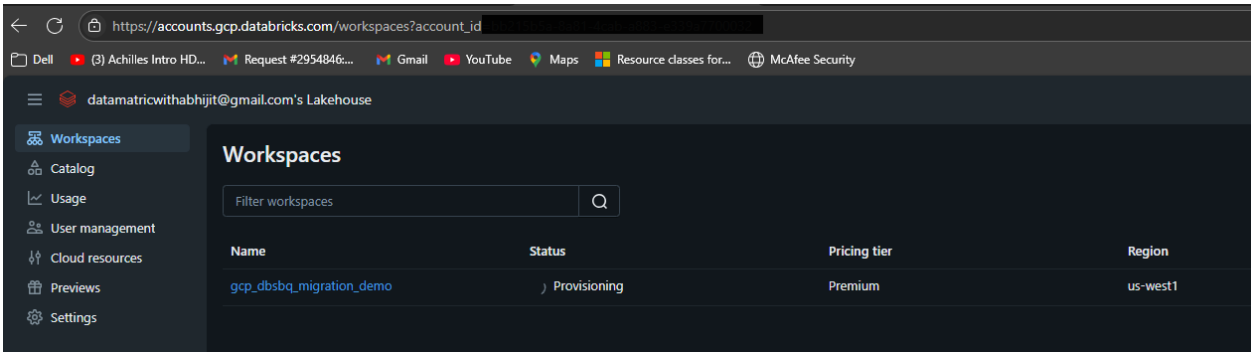





To create workspace, we need to fetch/copy GCP project-id.

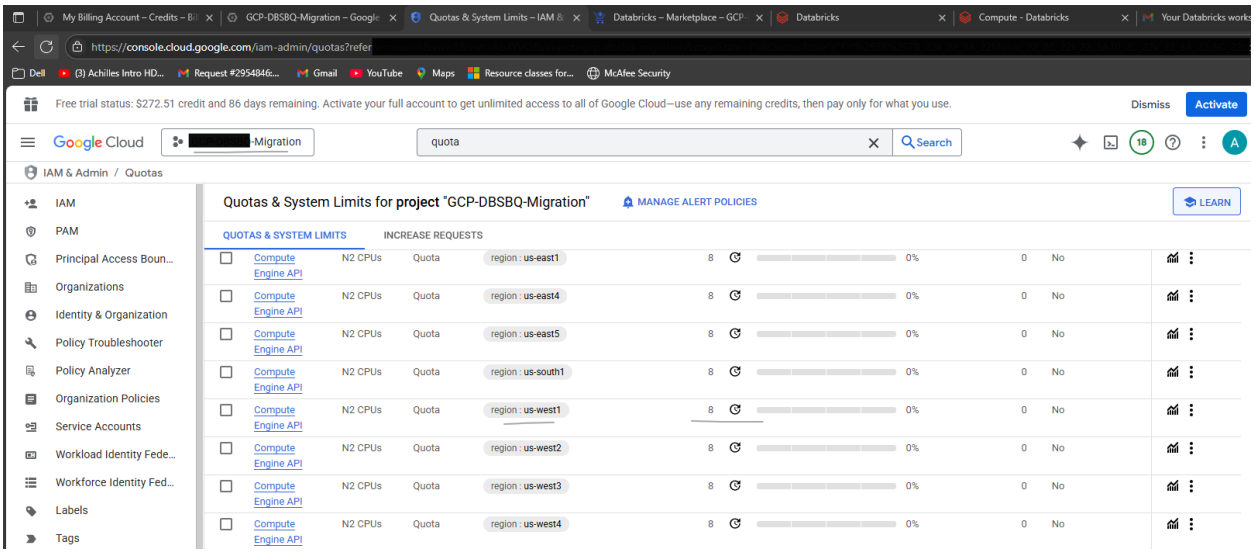




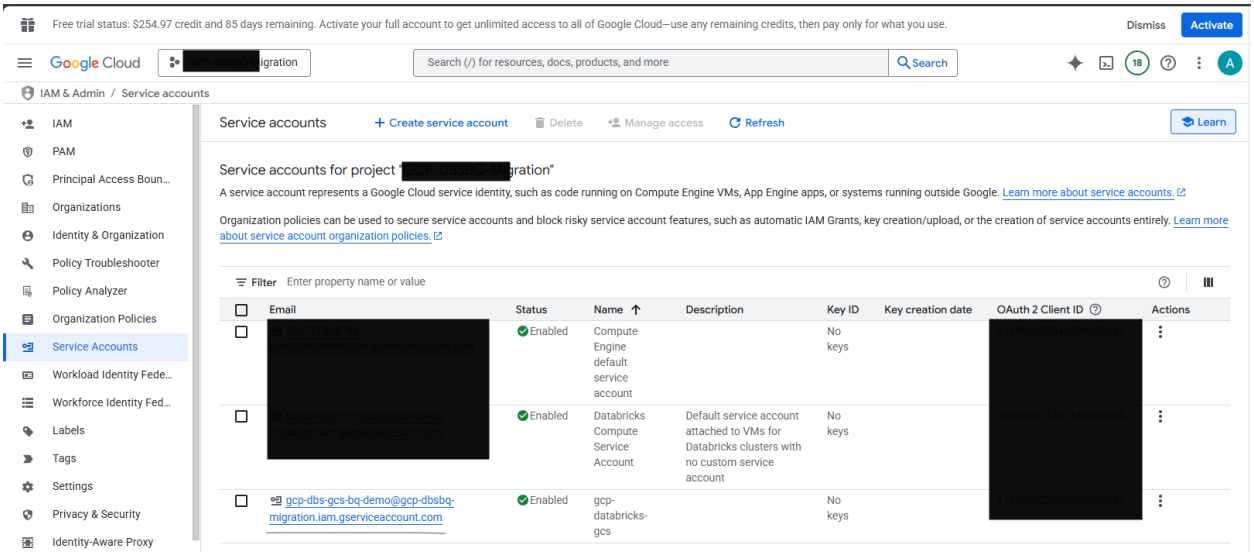




Create Databricks cluster - >

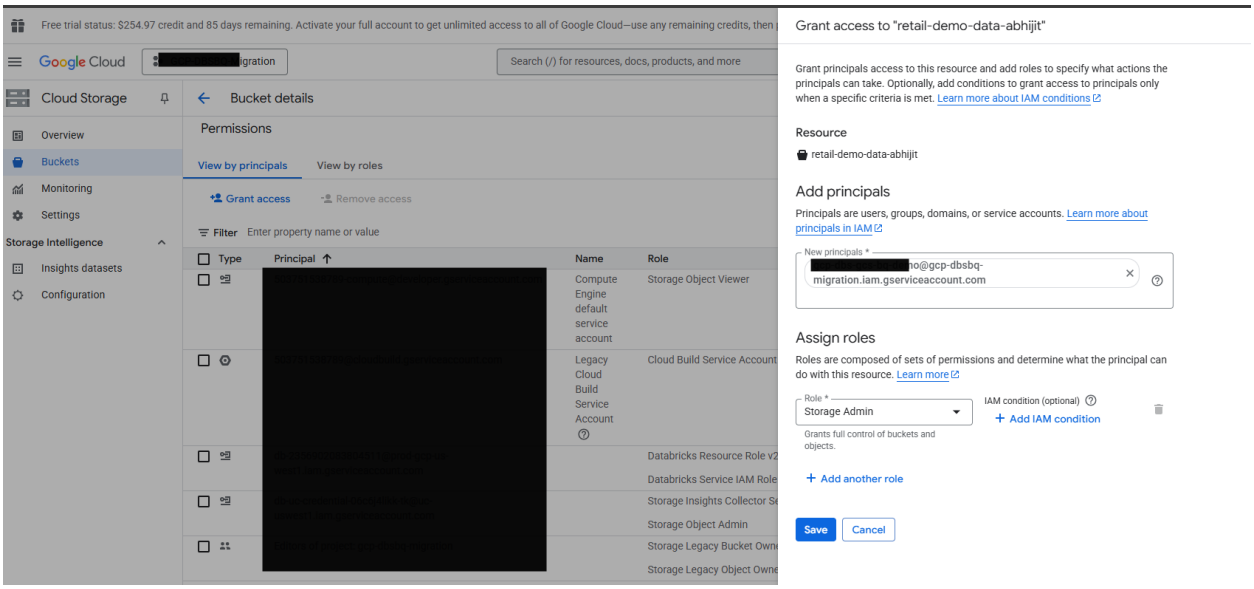
Check CPU availability



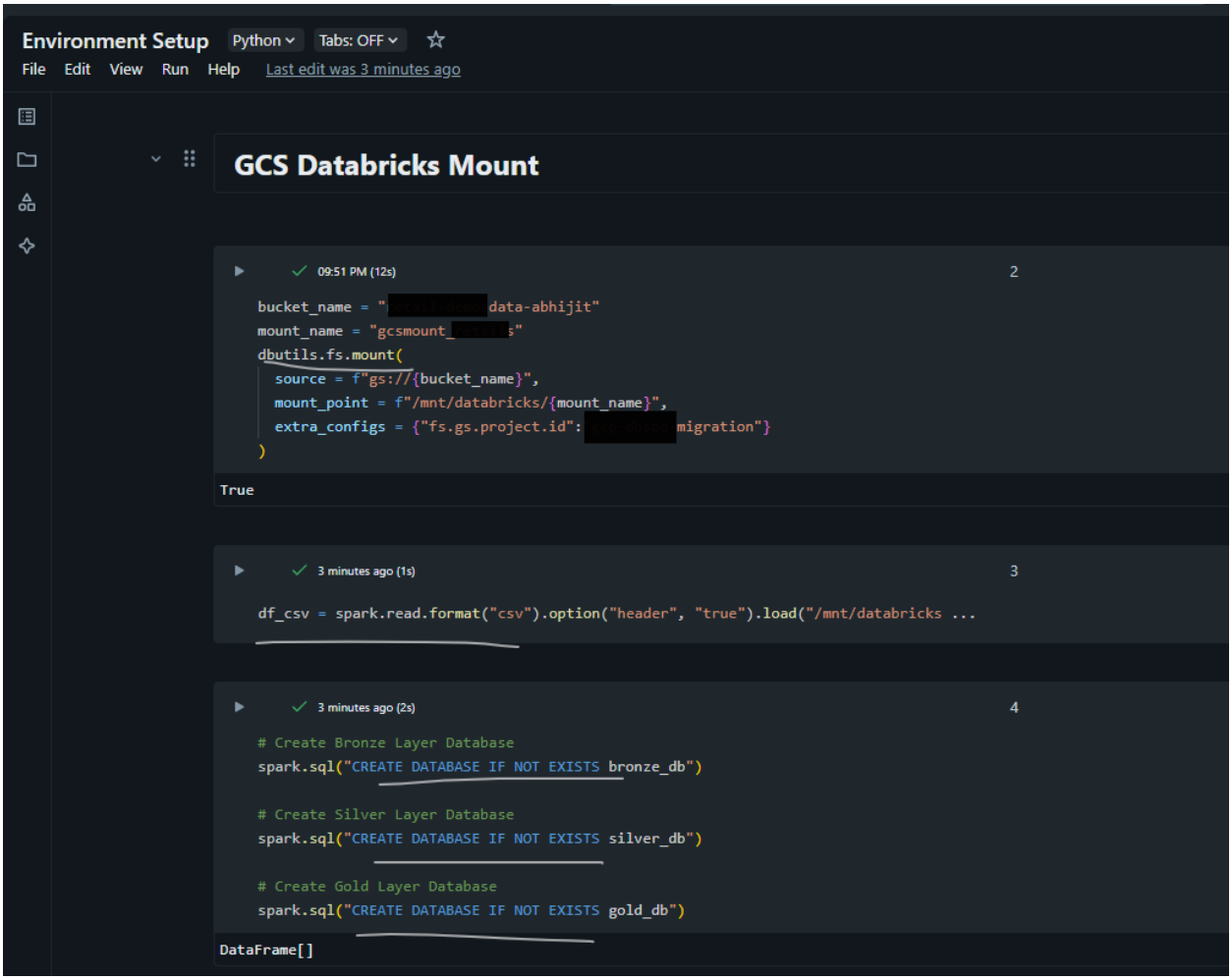
Now Databricks workspace is created. Next we will create a service account to access GCS bucket from Databricks.




Provide necessary permission to the service account in GCS bucket




Mount GCS buckets to access retails source data from GCP to Databricks. And create bronze, silver and gold databases in the Databricks default unity catalog.



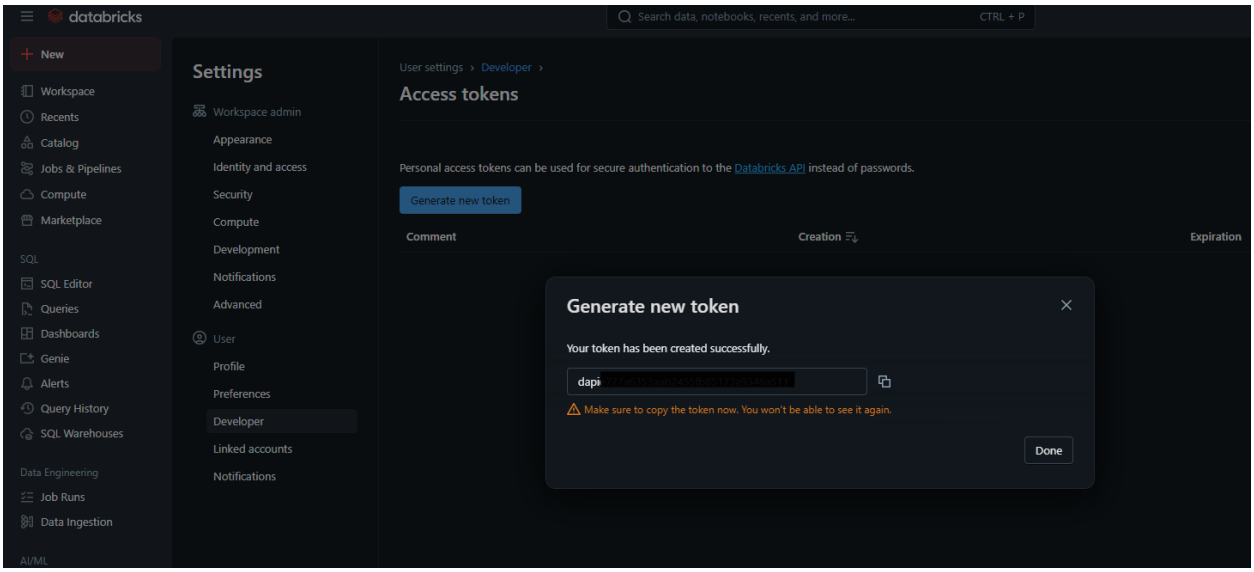

Agentic AI Use Case:

server_hostname="dbc-xxxxx.cloud.databricks.com", # Replace with your workspace hostname

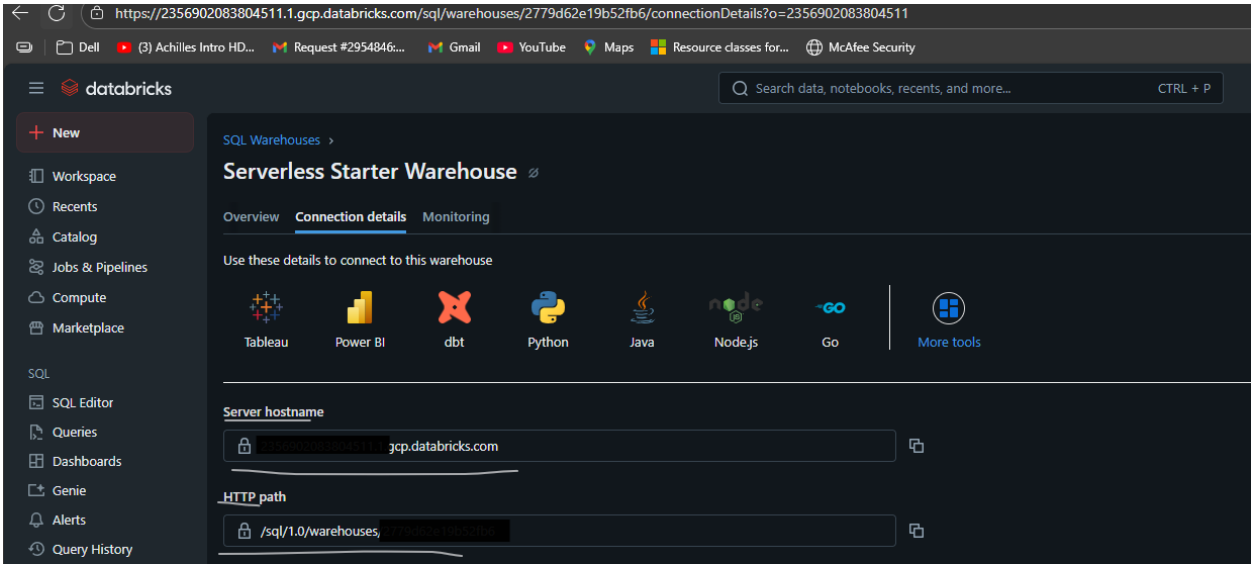
http_path="/sql/1.0/warehouses/xxxxxx", # Replace with your SQL warehouse path

access_token="dapi-xxxxxxxxxxxxxxxxx" # Use personal access token or notebook-scoped

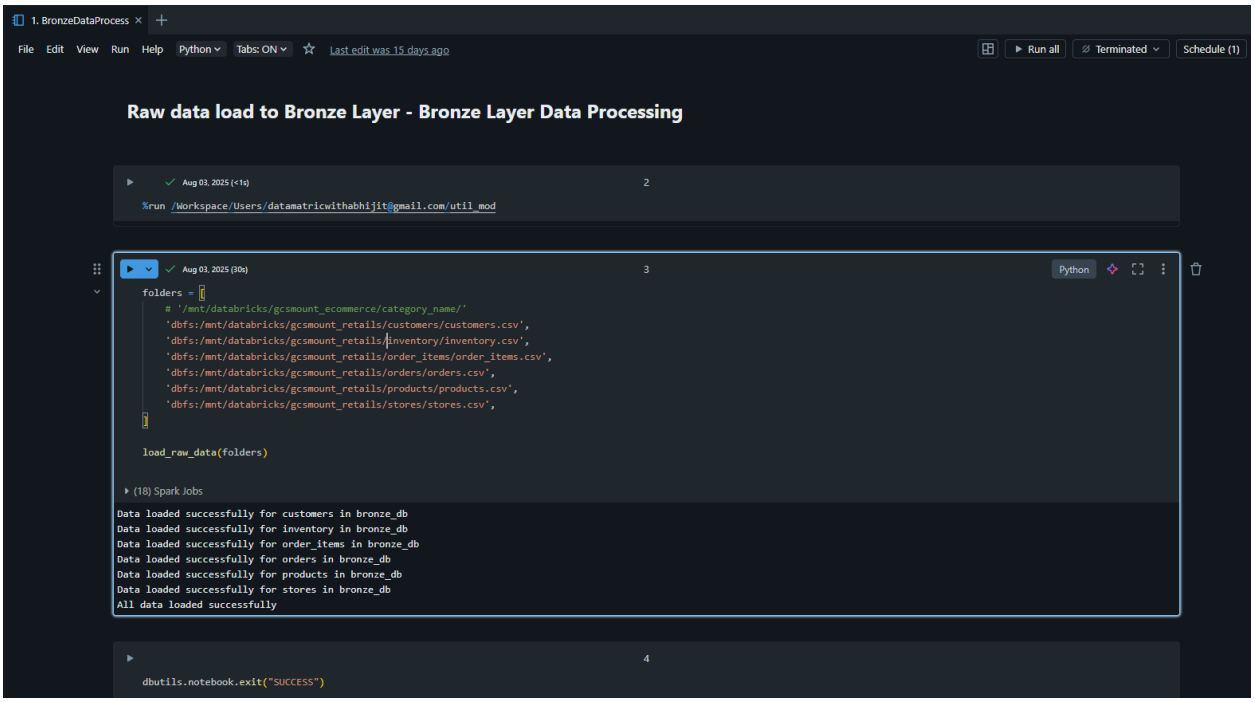
Create PAT:

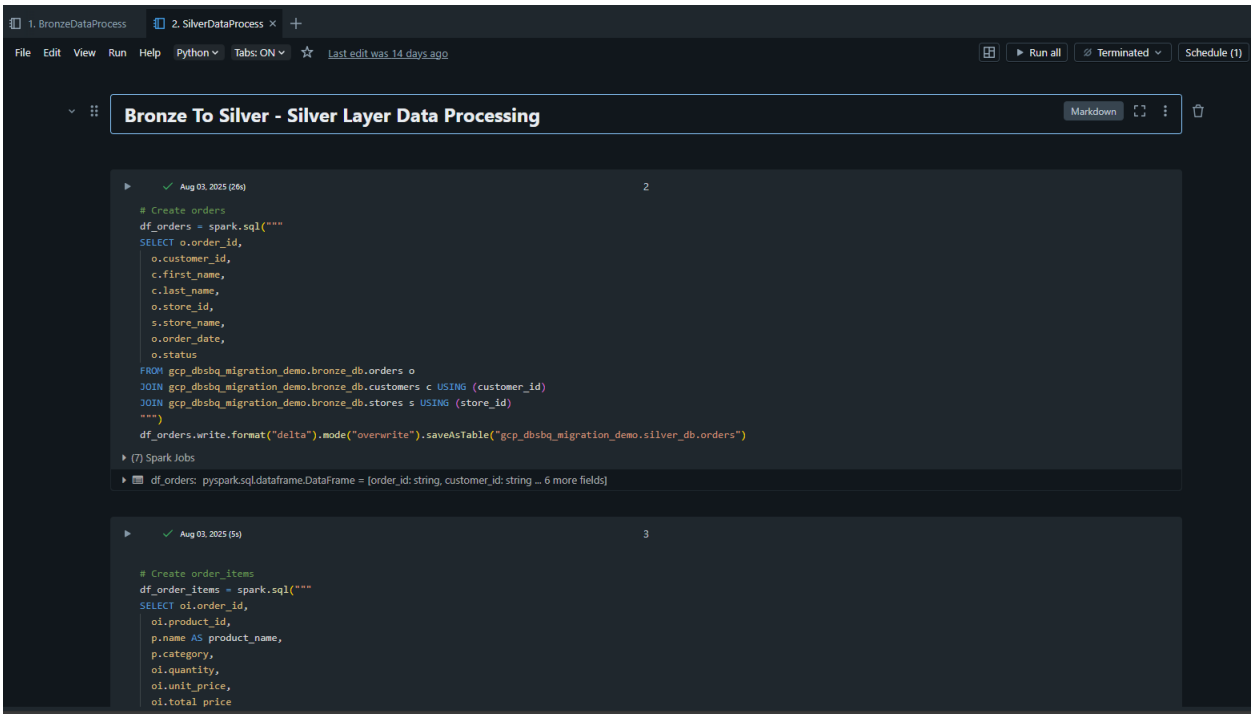


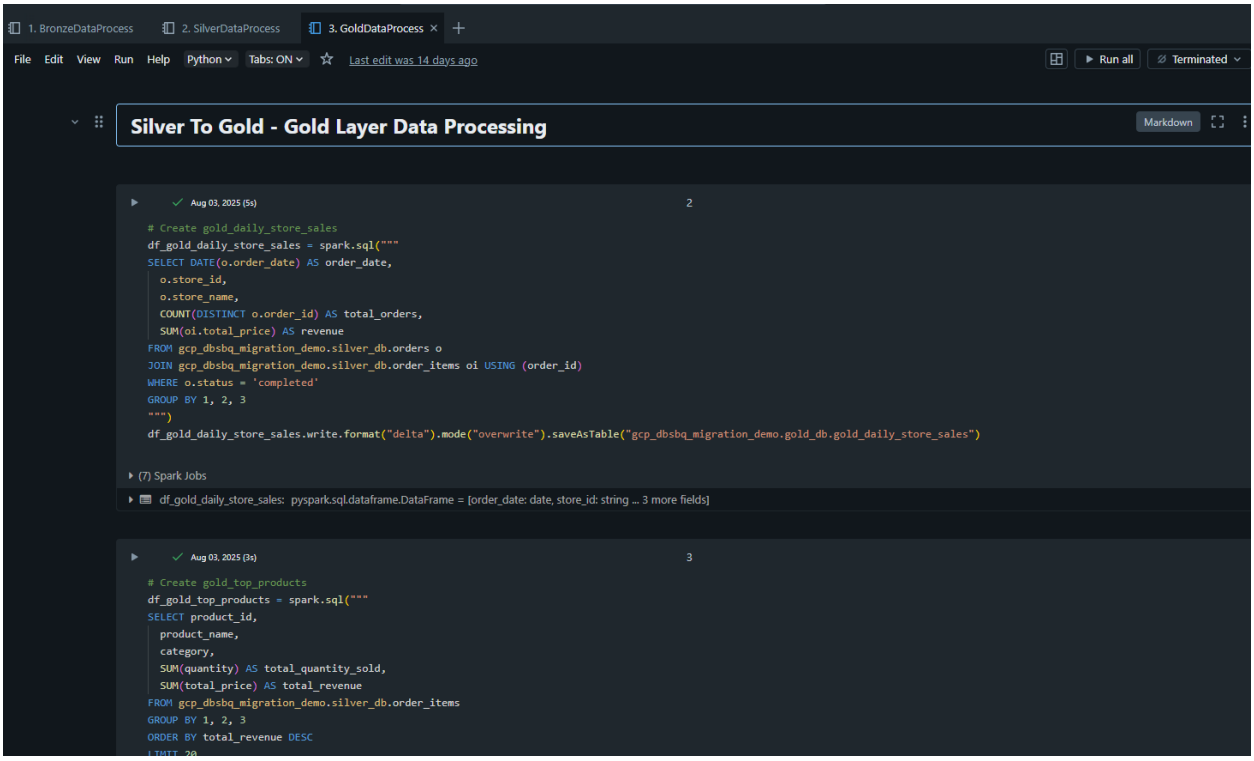

Get from SQL warehouse path:



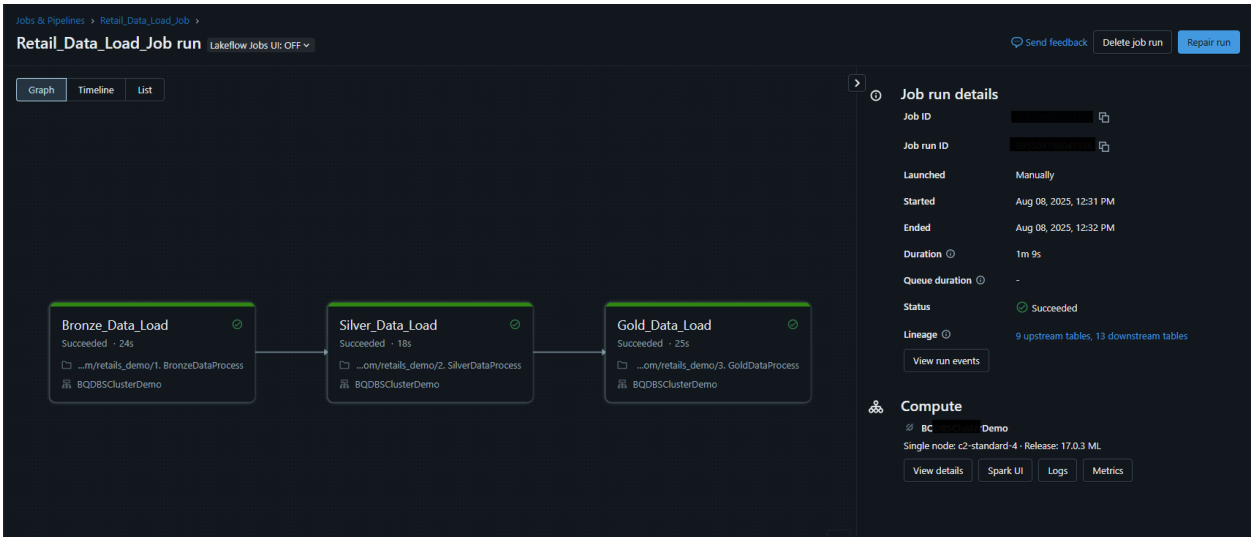

Create notebooks:


1. BronzeDataProcess
File Edit View Run Help Python Tabs: ON Last edit was 15 days ago
Run all Terminated Schedule (1)
Raw data load to Bronze Layer - Bronze Layer Data Processing
Aug 03, 2025 (<1s)
2
%run /Workspace/Users/datamatricwithabhijit@gmail.com/util_mod
Aug 03, 2025 (30s)
3
Python
folders = [
# '/mnt/databricks/gcsmount_ecommerce/category_name/'
'dbfs:/mnt/databricks/gcsmount_retails/customers/customers.csv',
'dbfs:/mnt/databricks/gcsmount_retails/inventory/inventory.csv',
'dbfs:/mnt/databricks/gcsmount_retails/order_items/order_items.csv',
'dbfs:/mnt/databricks/gcsmount_retails/orders/orders.csv',
'dbfs:/mnt/databricks/gcsmount_retails/products/products.csv',
'dbfs:/mnt/databricks/gcsmount_retails/stores/stores.csv',
]
load_raw_data(folders)
(18) Spark Jobs
Data loaded successfully for customers in bronze_db
Data loaded successfully for inventory in bronze_db
Data loaded successfully for order_items in bronze_db
Data loaded successfully for orders in bronze_db
Data loaded successfully for products in bronze_db
Data loaded successfully for stores in bronze_db
All data loaded successfully
4
dbutils.notebook.exit("SUCCESS")

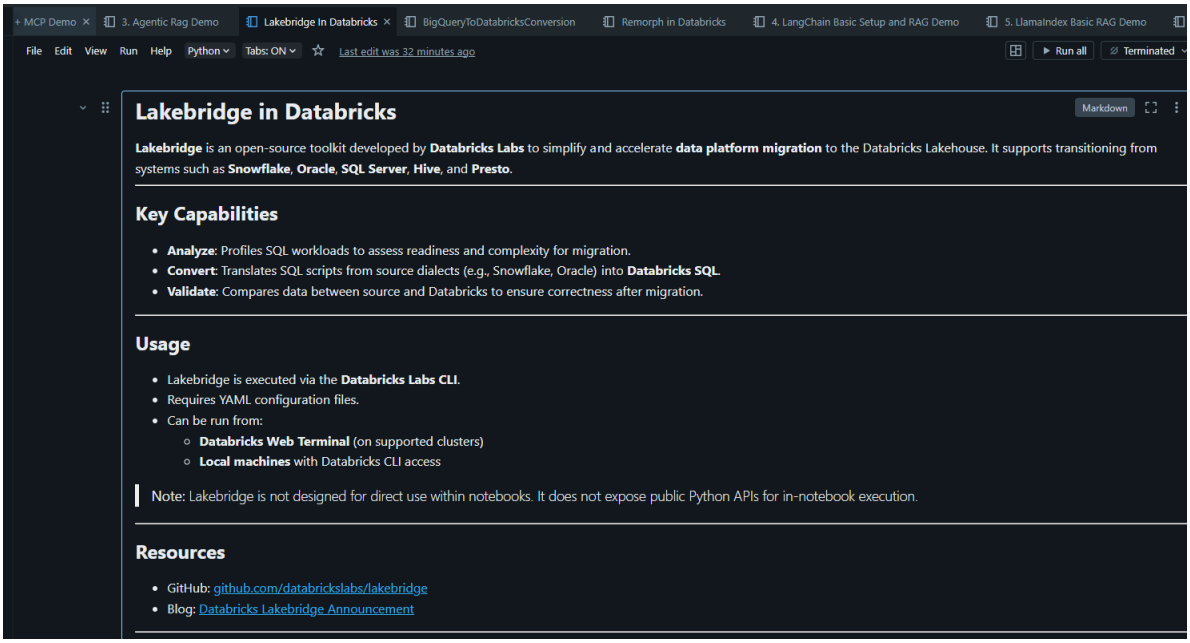

1. BronzeDataProcess
2. SilverDataProcess
File Edit View Run Help Python Tabs: ON Last edit was 14 days ago
Run all Terminated Schedule (1)
Bronze To Silver - Silver Layer Data Processing
Markdown
Aug 03, 2025 (26s)
2
# Create orders
df_orders = spark.sql("""
SELECT o.order_id,
o.customer_id,
c.first_name,
c.last_name,
o.store_id,
s.store_name,
o.order_date,
o.status
FROM gcp_dbsbq_migration_demo.bronze_db.orders o
JOIN gcp_dbsbq_migration_demo.bronze_db.customers c USING (customer_id)
JOIN gcp_dbsbq_migration_demo.bronze_db.stores s USING (store_id)
""")
df_orders.write.format("delta").mode("overwrite").saveAsTable("gcp_dbsbq_migration_demo.silver_db.orders")
(7) Spark Jobs
df_orders: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 6 more fields]
Aug 03, 2025 (5s)
3
# Create order_items
df_order_items = spark.sql("""
SELECT oi.order_id,
oi.product_id,
p.name AS product_name,
p.category,
oi.quantity,
oi.unit_price,
oi.total price

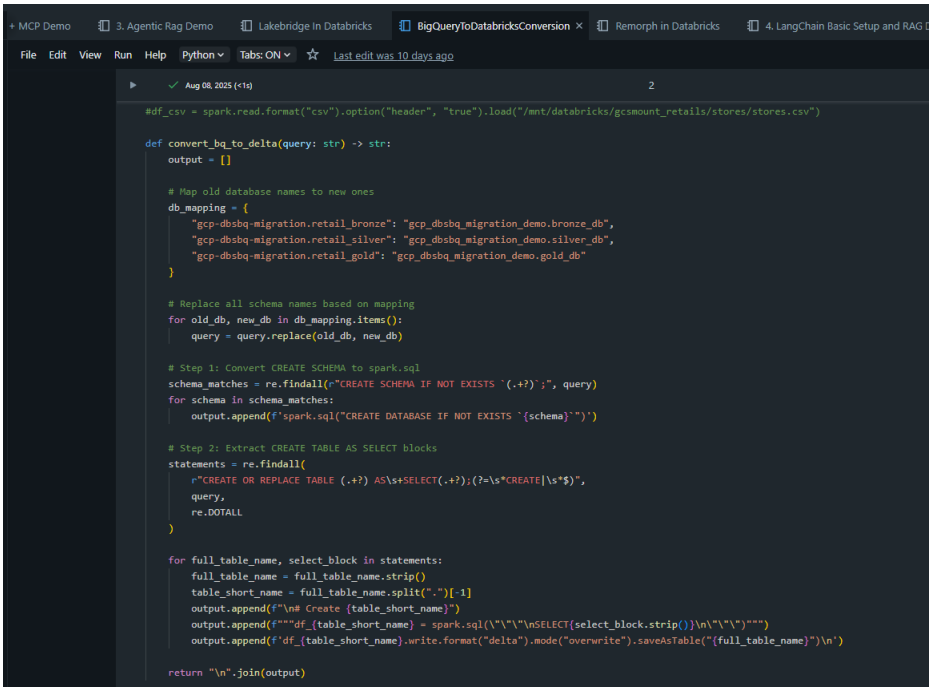


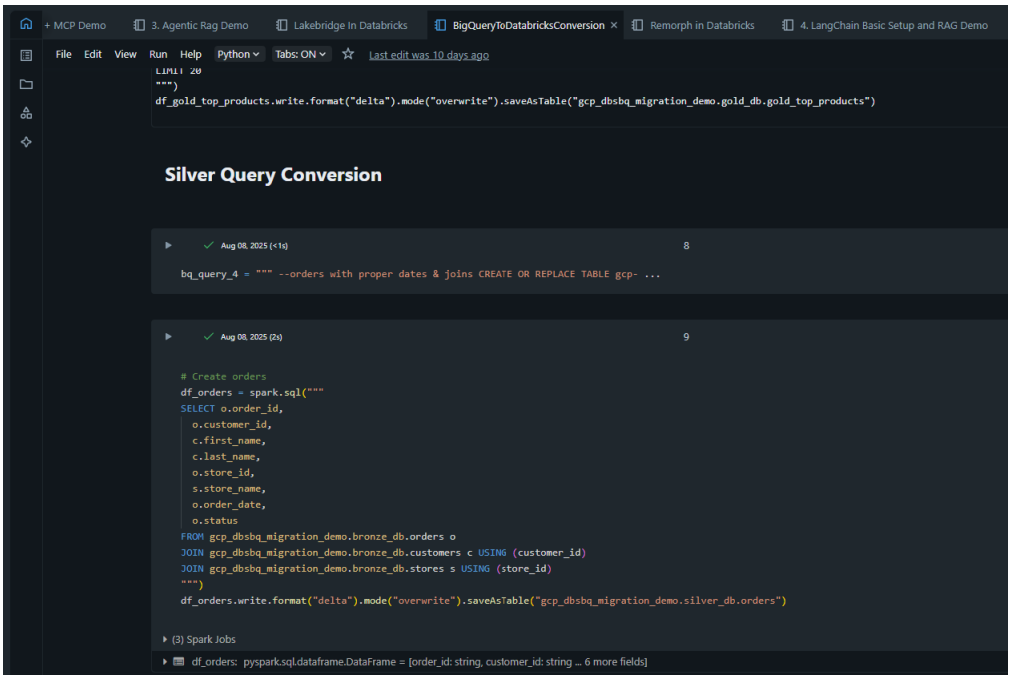

Workflow:

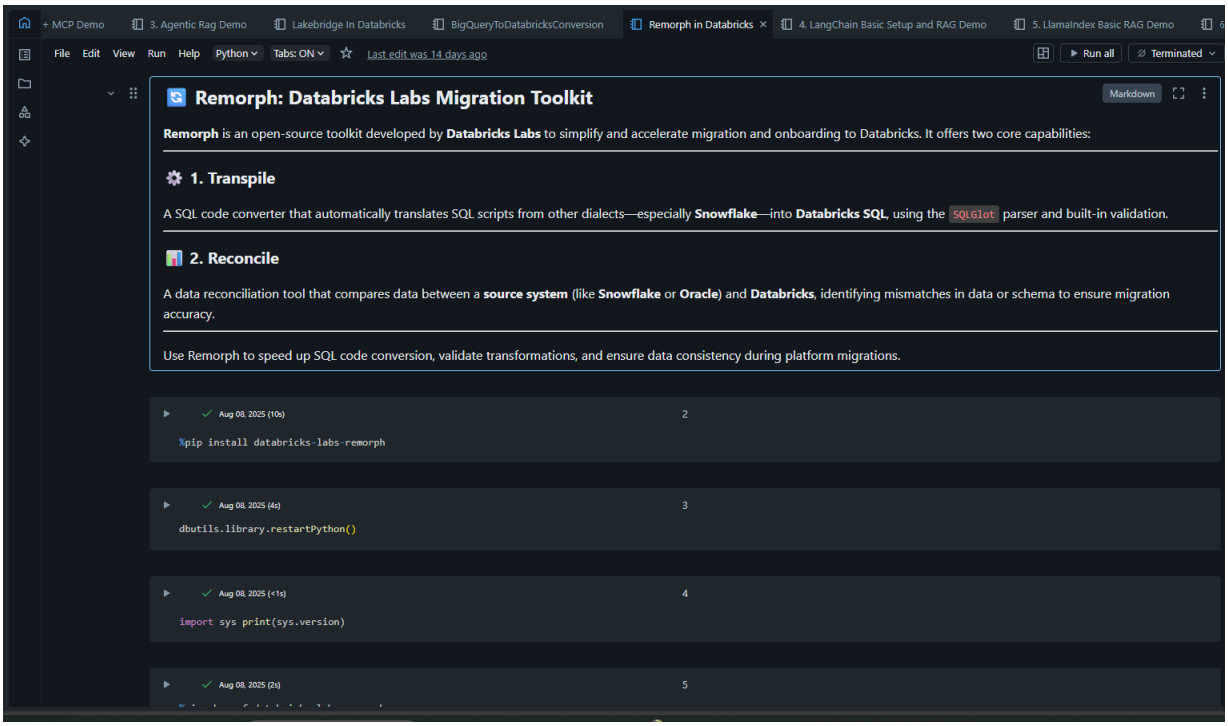



Lakebridge – Databricks Lab Accelerator:

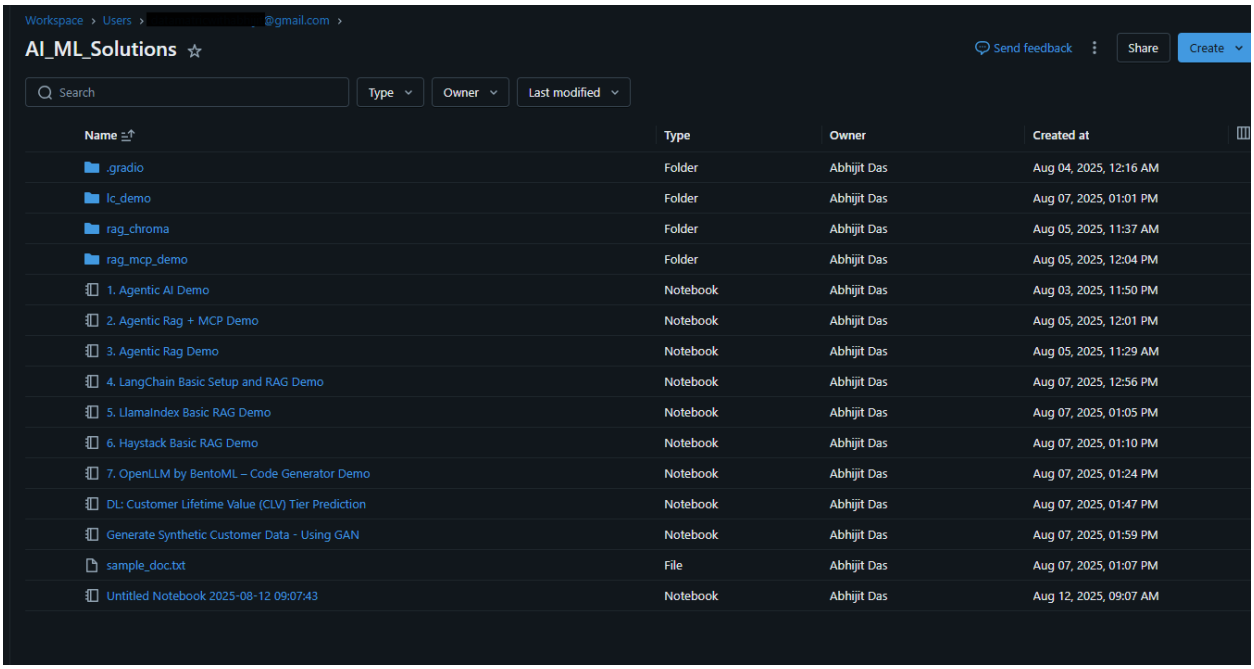

+ MCP Demo
3. Agentic Rag Demo
Lakebridge In Databricks
BigQueryToDatabricksConversion
Remorph in Databricks
4. LangChain Basic Setup and RAG Demo
5. LlamaIndex Basic RAG Demo
File Edit View Run Help Python Tabs: ON Last edit was 32 minutes ago
Run all
Terminated
Markdown
Lakebridge in Databricks
Lakebridge is an open-source toolkit developed by Databricks Labs to simplify and accelerate data platform migration to the Databricks Lakehouse. It supports transitioning from systems such as Snowflake, Oracle, SQL Server, Hive, and Presto.
Key Capabilities
Analyze: Profiles SQL workloads to assess readiness and complexity for migration.
Convert: Translates SQL scripts from source dialects (e.g., Snowflake, Oracle) into Databricks SQL.
Validate: Compares data between source and Databricks to ensure correctness after migration.
Usage
Lakebridge is executed via the Databricks Labs CLI.
Requires YAML configuration files.
Can be run from:
Databricks Web Terminal (on supported clusters)
Local machines with Databricks CLI access
Note: Lakebridge is not designed for direct use within notebooks. It does not expose public Python APIs for in-notebook execution.
Resources
GitHub: github.com/databrickslabs/lakebridge
Blog: Databricks Lakebridge Announcement

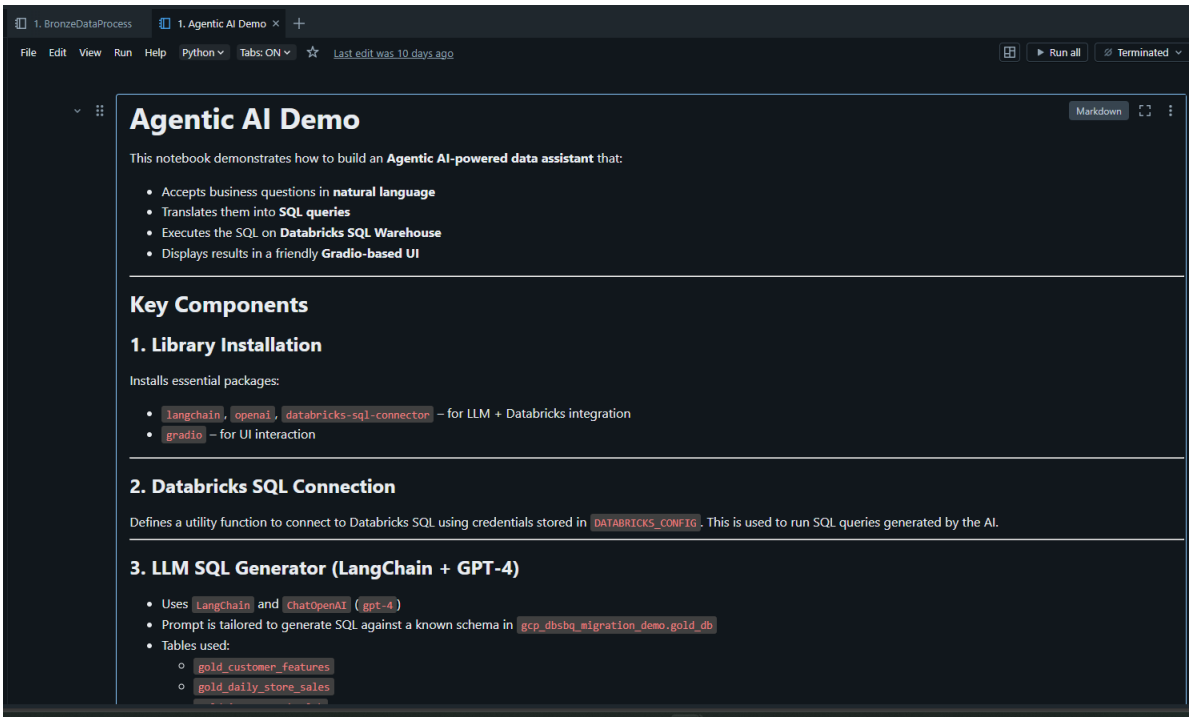

+ MCP Demo
3. Agentic Rag Demo
Lakebridge In Databricks
BigQueryToDatabricksConversion
Remorph in Databricks
4. LangChain Basic Setup and RAG
File Edit View Run Help Python Tabs: ON Last edit was 10 days ago
Aug 08, 2025 (<1s)
2
#df_csv = spark.read.format("csv").option("header", "true").load("/mnt/databricks/gcsmount_retails/stores/stores.csv")
def convert_bq_to_delta(query: str) -> str:
output = []
# Map old database names to new ones
db_mapping = {
"gcp-dbsbq-migration.retail_bronze": "gcp_dbsbq_migration_demo.bronze_db",
"gcp-dbsbq-migration.retail_silver": "gcp_dbsbq_migration_demo.silver_db",
"gcp-dbsbq-migration.retail_gold": "gcp_dbsbq_migration_demo.gold_db"
}
# Replace all schema names based on mapping
for old_db, new_db in db_mapping.items():
query = query.replace(old_db, new_db)
# Step 1: Convert CREATE SCHEMA to spark.sql
schema_matches = re.findall(r"CREATE SCHEMA IF NOT EXISTS `(.+?)`;", query)
for schema in schema_matches:
output.append(f'spark.sql("CREATE DATABASE IF NOT EXISTS `{schema}`")')
# Step 2: Extract CREATE TABLE AS SELECT blocks
statements = re.findall(
r"CREATE OR REPLACE TABLE (.+?) AS\s+SELECT(.+?);(?=\s*CREATE|\s*$)",
query,
re.DOTALL
)
for full_table_name, select_block in statements:
full_table_name = full_table_name.strip()
table_short_name = full_table_name.split(".")[-1]
output.append(f"\n# Create {table_short_name}")
output.append(f"""df_{table_short_name} = spark.sql(\"\"\"\nSELECT{select_block.strip()}\n\"\"\")""")
output.append(f'df_{table_short_name}.write.format("delta").mode("overwrite").saveAsTable("{full_table_name}")\n')
return "\n".join(output)

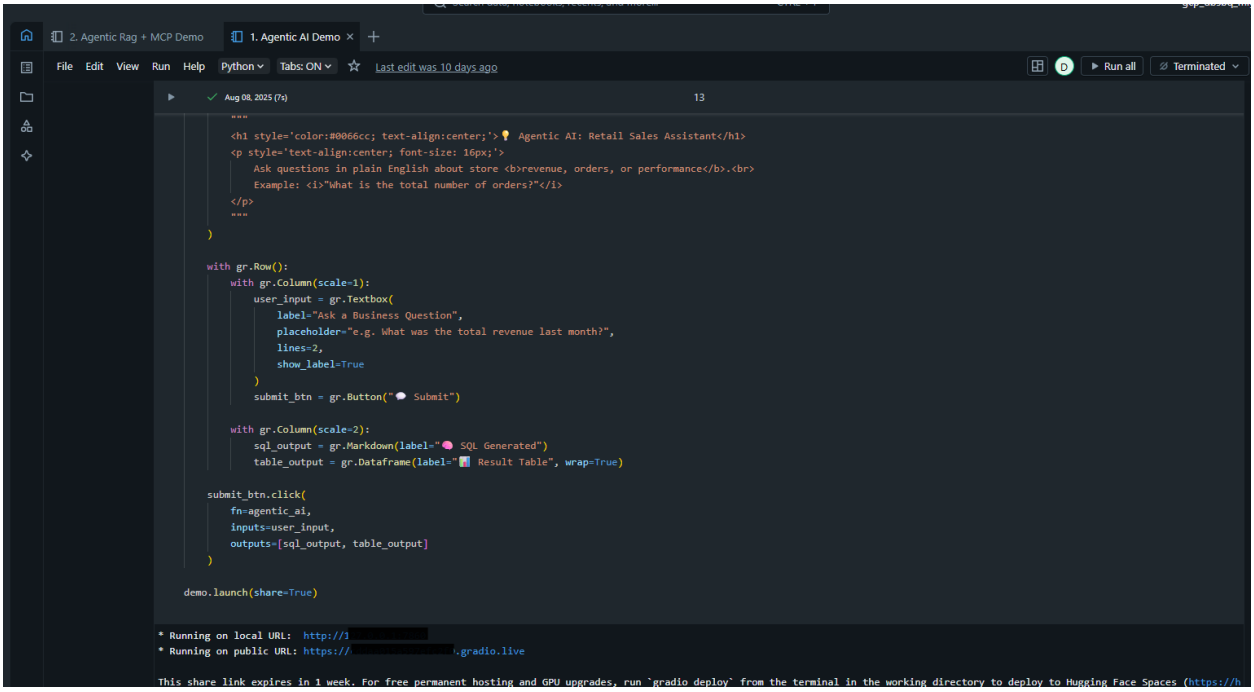

+ MCP Demo
3. Agentic Rag Demo
Lakebridge In Databricks
BigQueryToDatabricksConversion
Remorph in Databricks
4. LangChain Basic Setup and RAG Demo
File Edit View Run Help Python Tabs: ON Last edit was 10 days ago
LIMIT 20
"""
df_gold_top_products.write.format("delta").mode("overwrite").saveAsTable("gcp_dbsbq_migration_demo.gold_db.gold_top_products")
Silver Query Conversion
Aug 08, 2025 (<1s) 8
bq_query_4 = """ --orders with proper dates & joins CREATE OR REPLACE TABLE gcp- ...
Aug 08, 2025 (2s) 9
# Create orders
df_orders = spark.sql("""
SELECT o.order_id,
o.customer_id,
c.first_name,
c.last_name,
o.store_id,
s.store_name,
o.order_date,
o.status
FROM gcp_dbsbq_migration_demo.bronze_db.orders o
JOIN gcp_dbsbq_migration_demo.bronze_db.customers c USING (customer_id)
JOIN gcp_dbsbq_migration_demo.bronze_db.stores s USING (store_id)
""")
df_orders.write.format("delta").mode("overwrite").saveAsTable("gcp_dbsbq_migration_demo.silver_db.orders")
(3) Spark Jobs
df_orders: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 6 more fields]


+ MCP Demo
3. Agentic Rag Demo
Lakebridge In Databricks
BigQueryToDatabricksConversion
Remorph in Databricks
4. LangChain Basic Setup and RAG Demo
5. LlamaIndex Basic RAG Demo
File Edit View Run Help Python Tabs: ON Last edit was 14 days ago
Run all
Terminated
Remorph: Databricks Labs Migration Toolkit
Markdown
Remorph is an open-source toolkit developed by Databricks Labs to simplify and accelerate migration and onboarding to Databricks. It offers two core capabilities:
1. Transpile
A SQL code converter that automatically translates SQL scripts from other dialects—especially Snowflake—into Databricks SQL, using the SQLGlot parser and built-in validation.
2. Reconcile
A data reconciliation tool that compares data between a source system (like Snowflake or Oracle) and Databricks, identifying mismatches in data or schema to ensure migration accuracy.
Use Remorph to speed up SQL code conversion, validate transformations, and ensure data consistency during platform migrations.
Aug 08, 2025 (10s) 2
%pip install databricks-labs-remorph
Aug 08, 2025 (4s) 3
dbutils.library.restartPython()
Aug 08, 2025 (<1s) 4
import sys print(sys.version)
Aug 08, 2025 (2s) 5

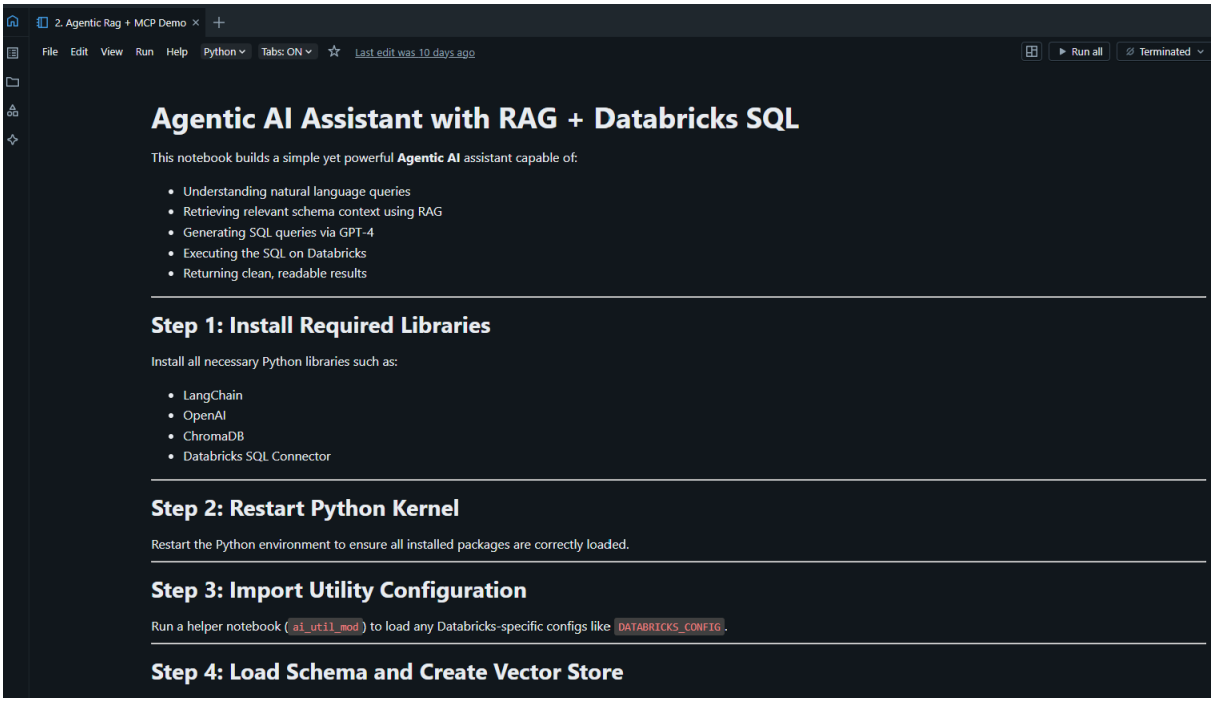

Agentic AI Introduction:

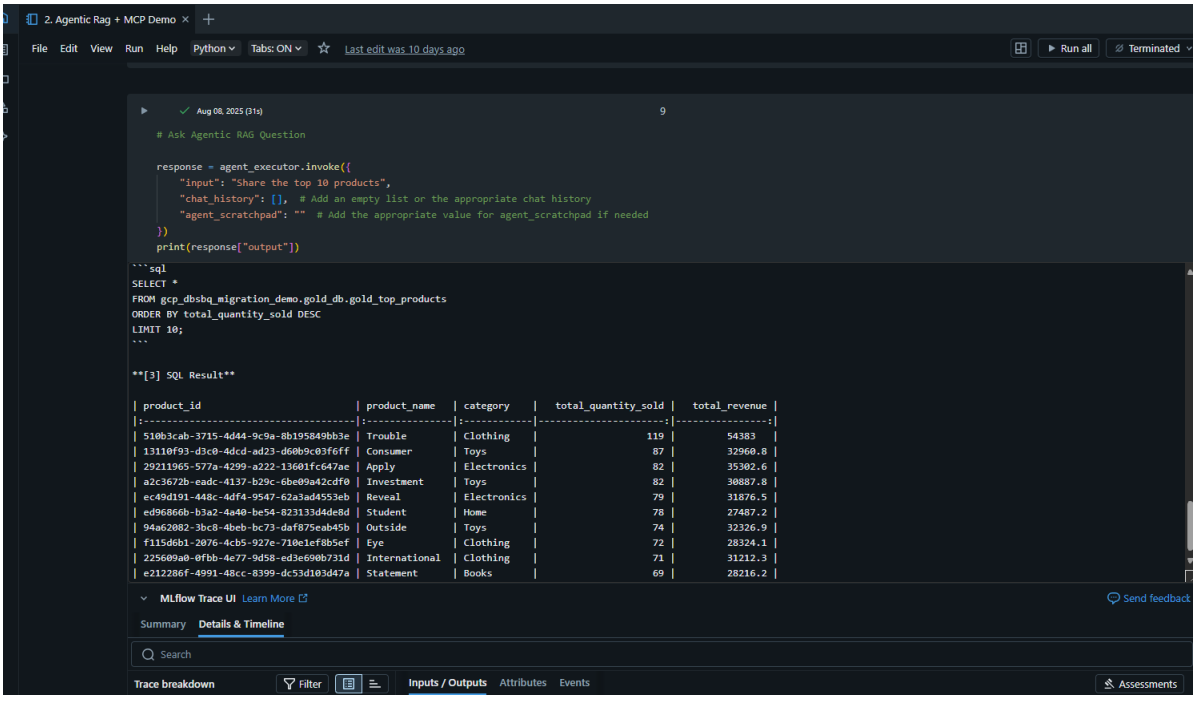




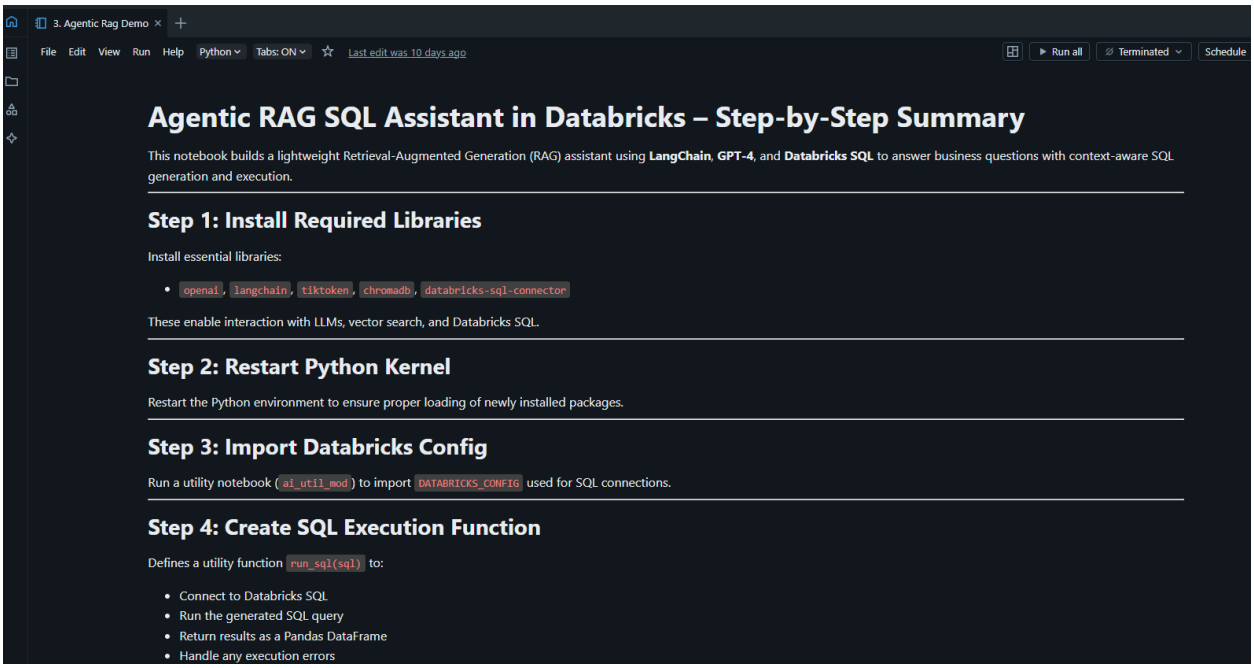



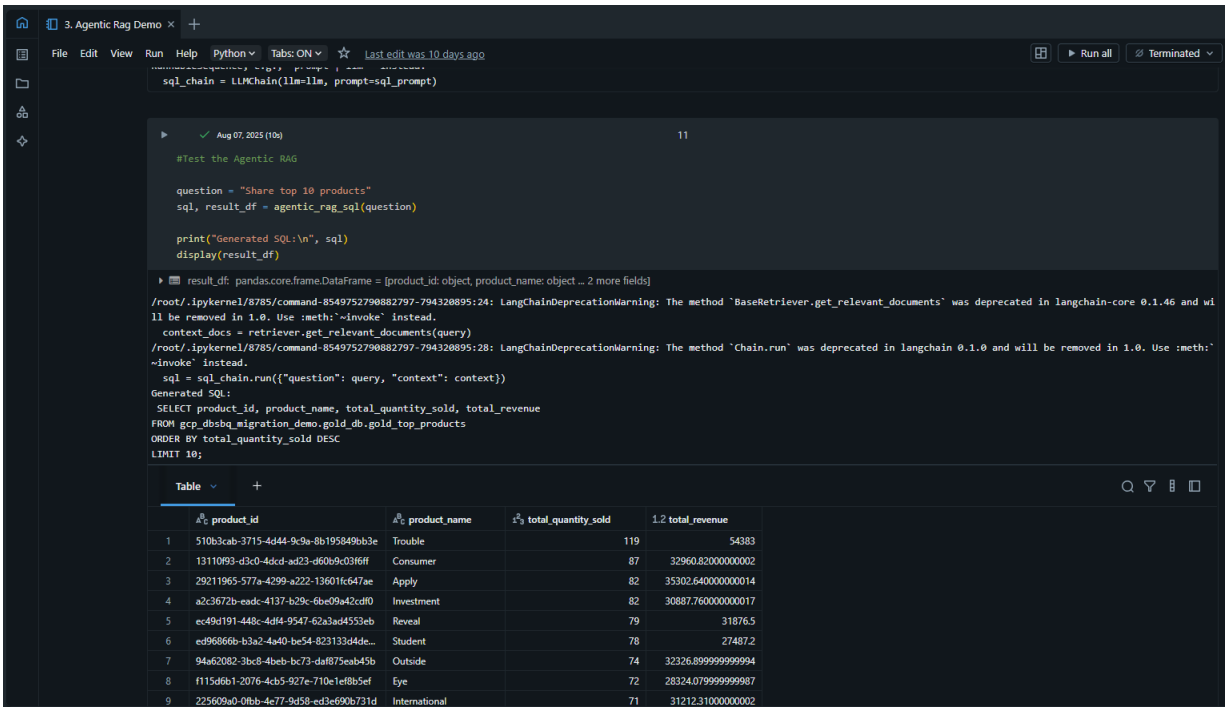

Agentic AI + RAG + MCP




Agentic AI + RAG


3. Agentic Rag Demo
File Edit View Run Help Python Tabs: ON Last edit was 10 days ago
Run all Terminated Schedule
Agentic RAG SQL Assistant in Databricks – Step-by-Step Summary
This notebook builds a lightweight Retrieval-Augmented Generation (RAG) assistant using LangChain, GPT-4, and Databricks SQL to answer business questions with context-aware SQL generation and execution.
Step 1: Install Required Libraries
Install essential libraries:
openai, langchain, tiktoken, chromadb, databricks-sql-connector
These enable interaction with LLMs, vector search, and Databricks SQL.
Step 2: Restart Python Kernel
Restart the Python environment to ensure proper loading of newly installed packages.
Step 3: Import Databricks Config
Run a utility notebook (ai_util_mod) to import DATABRICKS_CONFIG used for SQL connections.
Step 4: Create SQL Execution Function
Defines a utility function run_sql(sql) to:
Connect to Databricks SQL
Run the generated SQL query
Return results as a Pandas DataFrame
Handle any execution errors


3. Agentic Rag Demo
File Edit View Run Help Python Tabs: ON Last edit was 10 days ago
Run all Terminated
sql_chain = LLMChain(llm=llm, prompt=sql_prompt)
Aug 07, 2025 (10s)
11
#Test the Agentic RAG
question = "Share top 10 products"
sql, result_df = agentic_rag_sql(question)
print("Generated SQL:\n", sql)
display(result_df)
result_df: pandas.core.frame.DataFrame = [product_id: object, product_name: object ... 2 more fields]
/root/.ipykernel/8785/command-8549752790882797-794320895:24: LangChainDeprecationWarning: The method `BaseRetriever.get_relevant_documents` was deprecated in langchain-core 0.1.46 and will be removed in 1.0. Use :meth:`~invoke` instead.
context_docs = retriever.get_relevant_documents(query)
/root/.ipykernel/8785/command-8549752790882797-794320895:28: LangChainDeprecationWarning: The method `Chain.run` was deprecated in langchain 0.1.0 and will be removed in 1.0. Use :meth:`~invoke` instead.
sql = sql_chain.run({"question": query, "context": context})
Generated SQL:
SELECT product_id, product_name, total_quantity_sold, total_revenue
FROM gcp_dbsbq_migration_demo.gold_db.gold_top_products
ORDER BY total_quantity_sold DESC
LIMIT 10;
Table
product_id product_name total_quantity_sold total_revenue
1 510b3cab-3715-4d44-9c9a-8b195849bb3e Trouble 119 54383
2 13110f93-d3c0-4dcd-ad23-d60b9c03f6ff Consumer 87 32960.82000000002
3 29211965-577a-4299-a222-13601fc647ae Apply 82 35302.640000000014
4 a2c3672b-eadc-4137-b29c-6be09a42cdf0 Investment 82 30887.760000000017
5 ec49d191-448c-4df4-9547-62a3ad4553eb Reveal 79 31876.5
6 ed96866b-b3a2-4a40-be54-823133d4de... Student 78 27487.2
7 94a62082-3bc8-4beb-bc73-daf875eab45b Outside 74 32326.899999999994
8 f115d6b1-2076-4cb5-927e-710e1ef8b5ef Eye 72 28324.079999999987
9 225609a0-0fbb-4e77-9d58-ed3e690b731d International 71 31212.31000000002